AD 748301

SERIES	BASE NO. / VOL	REISSUE
TM -	3628 /008	/00

TECHNICAL MEMORANDUM

(TM Series)

The work reported herein was supported by the Advanced Research Projects Agency of the Department of Defense under contract DAHC15-67-C-0149.

COMPUTER-ASSISTED PLANNING

Semiannual Technical Summary Report
to the Director,
Advanced Research Projects Agency

for the period
16 September 1970 to 15 March 1971

C. Weissman, Project Director

SYSTEM
DEVELOPMENT
CORPORATION
2500 COLORADO AVE.
SANTA MONICA
CALIFORNIA
90406

The views and conclusions contained in this document are those of the author and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Advanced Research Projects Agency or the U.S. Government.

Reproduced by
NATIONAL TECHNICAL
INFORMATION SERVICE
U S Department of Commerce
Springfield VA 22151

~~This document has not been cleared for open publication~~

A-1159 (5/68)

APPROVED FOR PUBLIC RELEASE
DISTRIBUTION UNLIMITED

UNCLASSIFIED

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author) System Development Corporation Santa Monica, California	2a. REPORT SECURITY CLASSIFICATION Unclassified 2b. GROUP

3. REPORT TITLE

COMPUTER-ASSISTED PLANNING: Semiannual Technical Summary Report to the Director, Advanced Research Projects Agency, for the Period 16 September 1970 to 15 March 1971

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)

Technical Semiannual 16 September 1970 to 15 March 1971

5. AUTHOR(S) (First name, middle initial, last name)

Clark Weissman (Project Director)

6. REPORT DATE 15 April 1970	7a. TOTAL NO. OF PAGES 58	7b. NO. OF REFS 0
8a. CONTRACT OR GRANT NO. DAHC15 67 C 0149 b. PROJECT NO. ARPA Order #1327 c. d.	9a. ORIGINATOR'S REPORT NUMBER(S) TM-3628/008/00 9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report) NONE	

10. DISTRIBUTION STATEMENT

~~For official use only~~

[illegible]

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY

13. ABSTRACT

This report describes the first six months of a program of applied research and development whose purpose is to explore the practical implications and potential uses of computer technology in comprehensive military planning. The program, called Computer-Assisted Planning, is building on earlier work in Computer-Aided Command that focused on developing a prototype military computer utility (the ADEPT-50 system) and on using that facility as a tool for exploring the planning needs of military commanders.

The immediate goal of the Computer-Assisted Planning program is to investigate the impact of improved communications on the procurement and use of computers by Department of Defense planners. "Communications," in this context, includes both communication between the planner and his computer resources, and communication among computer facilities, especially those that are elements of a computer network. The program has two major objectives. The first is to provide military planners with models and procedures that will assist them both in strategic and tactical planning and in planning the acquisition and use of computation and communications resources. The second is to develop technologies and procedures that will enable planners to interface with their computers directly, through ordinary language and notational systems.

Progress is described in three major areas of study: (1) Computation and Communication Tradeoffs Study (CACTOS); (2) Natural Computer Input/Output; and (3) Systems Research and Interactive Systems. Plans for work during the remaining six months of the contract period are presented.

DD FORM 1 NOV 65 1473

UNCLASSIFIED

Security Classification

14 KEY WORDS	LINK A ROLE	LINK A WT	LINK B ROLE	LINK B WT	LINK C ROLE	LINK C WT
ADEPT Time-Sharing System						
ARPA Network						
Compiler Optimization						
Computer-Assisted Planning						
Computer Networks						
CONVERSE System						
Distributed Data Management						
Distributed Intelligence						
English Data Management						
Graphic Input/Output						
Graph Theory						
Man-Machine Synergy						
Problem Solving						
Voice Input/Output						

(Page iv Blank)

TABLE OF CONTENTS

PAGE

1. OVERVIEW 1

1.1 Goals and Objectives 1
1.2 Program Highlights 1
1.3 Organization of Report 4

2. COMPUTATION AND COMMUNICATION TRADEOFF STUDY (CACTOS) 5

2.1 Model Development Project 6
2.2 Data Base Development Project 12
2.3 Staff 14
2.4 Documentation 14

3. NATURAL COMPUTER INPUT/OUTPUT 15

3.1 CONVERSE: An English Data Management System 15
3.2 Graphic Input/Output 24
3.3 Voice Input/Output 29
3.4 Staff 34
3.5 Documentation 35

4. SYSTEMS RESEARCH 36

4.1 Networks 36
4.2 Graph-Meta 40
4.3 Staff 46
4.4 Documentation 46

5. INTERACTIVE SYSTEMS 47

5.1 Problem Solving and Learning by Man Machine Teams ... 47
5.2 Time-Sharing 51
5.3 LISP Extensions 56
5.4 Staff 57
5.5 Documentation 58

LIST OF FIGURES

		PAGE
Figure 2-1	ARPA Network (with modifications)	9
Figure 2-2	Data Input for ARPA Network	10
Figure 3-1	Question Answering in CONVERSE V-Ø	18
Figure 3-2	Surface (top) and Deep (bottom) Structures Produced by the Natural Language Compiler	19
Figure 3-3	Dictionary Building	26 & 27
Figure 3-4	TEST Mode	28
Figure 3-5	Voice I/O Project Computer Configuration and Other Computing Resources	31
Figure 3-6	Input Hardware for the Voice I/O Laboratory	32
Figure 5-1	Programmable Controller and Associated Equipment ...	54

1. OVERVIEW

This report describes the first six months (from 16 September 1970 to 15 March 1971) of a program of applied research and development whose purpose is to explore the practical implications and potential uses of computer technology in comprehensive military planning. The program, called Computer-Assisted Planning, is building on earlier work in Computer-Aided Command that was focused on developing a prototype military computer utility (the ADEPT-50 system) and on using that facility as a tool for exploring the planning needs of military commanders.

1.1 GOALS AND OBJECTIVES

Research covered by this contract is directed toward the long-range goal of substantially improving understanding of the strategic planning process, with the eventual objective of embodying that understanding in an experimental, prototype computer-based planning system. The system envisioned would integrate old and new technologies, permitting teams of Department of Defense (DoD) planners to interact with each other, with computer-based data bases, and with analysis tools via natural communications in order to achieve planning objectives more rapidly or with higher quality.

The immediate goal of the Computer-Assisted Planning (CAP) program is to investigate the impact of improved communications on the use of computers by DoD planners. "Communications," in this context, includes both communication between the planner and his computer resources, and communication among computer facilities, especially those that are elements of a computer network. The CAP program has three major objectives: (1) to provide military planners with models and procedures that will assist them both in strategic and tactical planning and in planning the acquisition and use of computation and communications resources; (2) to develop technologies and procedures that will enable planners to interface with their computers directly, through ordinary language and notational systems; and, (3) to develop the computer systems technologies needed to construct future planning systems.

1.2 PROGRAM HIGHLIGHTS

The work reported herein covers three areas: Computation and Communication Tradeoff Studies (CACTOS); Natural Computer Input and Output; and Systems Research.

CACTOS

CACTOS is examining the 1975-80 DoD requirements for computation and communication resources, with particular attention to the tradeoffs between concentrated and distributed computation power. Work divides into development of a computer network analysis model and the definition and construction of a

real-world planning data base on which to exercise the model. Progress was made in both areas with the selection of the Marine Corps Personnel Network system for study. This non-trivial, non-classified system provides the practical context for model and data base construction, which is in progress. Earlier, a prototype network analysis model was built and tested against the ARPA Network both as it now exists and as it might be reconfigured. Interesting results are reported that suggest modifications to the ARPA Network topology.

Natural Computer Input and Output

This research is directed toward integrating the interface between man and machine to make their transactions as "human" as possible--through English, both by keyboard and by voice, and through two-dimensional mathematics and drawings.

CONVERSE version V-Ø, a significant milestone for an English-language Data Management System (EDMS), was demonstrated during this period. Version Ø is the first of a new series of such EDMS's. It translates into a data management language English questions of moderate complexity, using only surface-structure information and data from a small concept network. Successor versions V-1 and V-2, currently under development, employ new surface-structure and deep-structure parsers, a richer data base, a capability for recognizing and processing declarative and imperative sentences (in addition to interrogative sentences), and user-feedback and language-extension facilities.

Years of work in hand-printed character recognition and two-dimensional graphic input/output came together during this period with the impressive demonstration of The Adding Machine (TAM). Operationally, TAM can be viewed as a "reactive blackboard" on which the user prints arithmetic expressions that the computer reacts to by printing back correct evaluations. The level of accomplishment includes a powerful set of operators; fixed, floating-point, and array variables; looping; and both built-in and user-defined functions. Extensions to TAM to enable symbolic processing operators are being designed, with the goal of achieving a fully interactive computational language.

This past half year of our Voice I/O research was devoted to laboratory development and planning strategies for solving the continuous speech problem. We have selected an incremental approach leading to a Voice-CONVERSE system. Starting from a Vicens-Reddy (V-R) base, we are aiming toward an intermediate accomplishment--a Voice Data Management System (VDMS) of linguistic complexity matching such current data management systems as TDMS or DS/2.* Highlights of our program to date include the completion of

*TDMS and DS/2 are proprietary SDC data management systems.

our voice laboratory built around the Raytheon 704 computer for signal processing of acoustic information. The reimplementation of the V-R system is nearly complete, and significant analyses of the V-R algorithms and heuristics, not heretofore clearly understood, have been published. Extensions and improvements to V-R are continuing in a system called CWIPER--our base for future VDMS progress.

Systems Research

Our efforts in systems research are toward the practical realization of the computer-system technologies that a comprehensive CAP system will require. During this reporting period, our time-sharing system, ADEPT, was successfully moved from the IBM 360/50 computer to the IBM 360/67. The move aided our research significantly by providing a compatible, reliable executive, expanded to support faster terminals (up to 300-baud) and a LISP 1.5 system that has been expanded to 85 pages (approximately 348,000 bytes) of resident core memory.

A significant ARPA Network milestone was reached as this document went to press: a test--SDC to RAND--of interprocess communication. Complete network operation (TELNET) will be available this summer as we complete our time-sharing Network program (HOSTCSS) and retrofit existing routines to comply with recent HOST-to-HOST protocol changes. The distributed data base study has concluded that network data management can best be achieved through the integration of local Node Data Management Systems (NDMS) with a common network data management language and appropriate local interfaces. English is proposed as that common network language, and a single, central-node, CONVERSE-like translator system is being advanced to translate English queries into each NDMS form.

Theoretical work on the use of graphs in global program optimization was completed and documented this period as Part I of a two-part text entitled "A Mathematical Theory of Global Program Analysis." Part II, dealing with practical applications, is in preparation; the complete text will be finished during this contract year. A practical demonstration of this research is embodied in the construction of a benchmark FORTRAN IV compiler. Pass I of the three-pass compiler has been designed, coded, and is in check-out; passes II and III are in design.

Gaku, the model of man-machine cooperative problem solving, is coming to grips with the organizational problems needed to build a complex planning system. Gaku is evolving incrementally as a collection of rules written in the User-Adaptive Language (UAL) specifically designed for stating such complex problems. During the reporting period, major advances were made in the construction (in SDC LISP) of a basic UAL. A modestly sophisticated demonstration of man-machine interaction is now possible. To achieve the level of progress reported herein for both UAL and CONVERSE, a number of

improvements were required to SDC LISP. Most significant was the dramatic (quadrupling) of LISP user data space made possible by the 85-page LISP under ADEPT on the IBM 360/67.

1.3 ORGANIZATION OF REPORT

The body of this report describes in detail the projects devoted to the three main areas of investigation: Computation and Communication Tradeoff Studies (section 2), Natural Computer Input and Output (section 3), and Systems Research (sections 4 and 5). Each section includes a detailed description of the projects pursued, the professional staff, and the technical publications produced during the contract period. The goals, problems, successes, progress, and status for the past six months are described for each project.

2. COMPUTATION AND COMMUNICATION TRADEOFF STUDY (CACTOS)

The goal of the Computation and Communication Tradeoff Study is to determine specific DoD requirements for computation and communication networks on a regional, functional, and categorical basis for the 1975-80 time period. With the continuing advance of technology in computer hardware, software, and communications, it is important that an overall analysis be done of DoD requirements in these areas. Implicit in such an analysis are the tradeoffs between various characteristics of computer and communication networks, which must be examined several years prior to the procurement of equipment and facilities, and which must be related not only to cost and time, but to each other. Such a tradeoff study must be at least in part quantitative and be capable of wide usage so as to relate to specific network configurations within DoD.

To achieve this goal, the following objectives have been identified:

1. Determine amounts of data processed, stored, and retrieved, including response time and tradeoffs between security, vulnerability, throughput, reliability, cost, and time.

2. Construct analytic models for evaluating and modifying selected networks.

3. Perform tradeoff studies based, in part, on the results of the model analysis.

4. Validate the analysis using an existing military network.

5. Describe future technology in areas relating to computers and communications.

During the past six months, effort was spent in (1) delineating the descriptors of a network, (2) establishing the ingredients of the data base, and (3) selecting a validation DoD network. Effort was also directed at obtaining state-of-the-art forecasts of technology for central processing units, memories, operating systems, control and display devices (terminals), communication channels and services, modems, switches, concentrators, and software.

An analysis was conducted of what key computation and communication characteristics of networks would apply generally, be available from analyzing data, and permit the comparison of networks. As this was done, it became clear that two general efforts were needed. The first is that of developing and adapting a network model, and the second is that of selecting an appropriate data base for use in the initial analysis. These efforts are described separately in the following pages. It should be noted here that

the search for an appropriate data base resulted in the selection of the Marine Corps Personnel Network, a non-trivial, unclassified, and well-documented system. The Marine Corps is providing us with considerable data and cooperation.

2.1 MODEL DEVELOPMENT PROJECT

2.1.1 Progress

The model development had as its initial goal the construction of a prototype network analysis model. This has been achieved and is described below.

The model operates on line under TS/DMS* and ADEPT at SDC. It is programmed in FORTRAN IV and operates within the framework of the SDC program, DESIGNET.** The model inputs consist of network configurations (nodes and links), node and link characteristics (processing capabilities and costs), and the sizes, arrival rates, sources, and destinations of messages and jobs. The network configuration is entered either on line, by individual links or node connectivities, or by selection from a data base. A job-arrival-rate matrix is specified, and message sizes, job sizes, and link lengths may be given either specific matrices or average values for the net. Job processing rates are given for each node, and channel capacities may be specified either for each channel or for the net as a whole. (If a total channel capacity is specified, the model will compute an optimal allocation for each channel to match the message traffic.) Each of these inputs may be specified on line in a conversational mode or selected from a data base. The inputs to the model are saved and can be modified on line by the user, who can add or delete nodes and links, change traffic characteristics, and run the model in an iterative fashion to find optimal conditions.

The outputs from the model consist of a network analysis, a performance analysis, and, if desired, a traffic analysis for either nodes or links or both. The model will also produce a graph of any two performance variables (e.g., response time and cost) plotted against one another. The network analysis consists of a statement of the numbers of nodes and links, the link-to-node ratio (assuming full-duplex lines), the variance (an indication of the "clumpiness" of the network), the radius (the "shortest longest" path between any two nodes), the diameter (the shortest path between the two most distant nodes), the number of articulation points of order one (a point that, if deleted, would break the net into two subnets), and the number of

---

*TS/DMS is a commercial time-sharing system marketed by SDC.

**DESIGNET is a proprietary SDC software package for analyzing the optimum distribution of resources in an interconnected commodities network.

circuits of length three. The performance analysis consists of the average path length, the mean communication response time, the mean computation response time, the total response time, and the total cost. The traffic analysis gives traffic, capacity, delay time, and cost for each link, and traffic, delay, and cost for each node.

Network characteristics can be treated as variables and manipulated to determine the network performance and cost effects of changing conditions (e.g., anticipated growth or shifts in processing requirements), or to evaluate changes to existing or proposed networks. Some of the variables that may be manipulated for given network configurations and traffic densities are:

1. Load Conditions. The model probes areas of response sensitivity for various combinations of job and message arrival rates, job and message sizes, and traffic patterns.

2. Computation and Communication Capacities. The effects of manipulating node and link processing capacities can be evaluated. Relative bottlenecks and alternative routings can be created by adjusting capacities and by adding and deleting links and nodes.

3. Network Vulnerability. The network's vulnerability to accidental or deliberate destruction of nodes and links, and the corresponding degradation in performance, can be evaluated. Finding network weaknesses presents a clear challenge to network designers, as does minimizing degradation of performance under a hostile environment.

4. Network Topological Characteristics. The effects of changing such topological characteristics as the radius, diameter, link-to-node ratio, and variance can be evaluated. Determining direct relationships between topological characteristics and performance would lead to principles for creating efficient networks.

5. Distributed Intelligence. The effects of the relative centralization versus the relative dispersion of logic (computing capacity) and information stores can be evaluated. The effects of advancing technology--for example, drastic reductions in logic, storage, and transmission costs--on the relative cost-effectiveness of different distributions of intelligence is a major tradeoff being considered in CACTOS.

There are certain assumptions and capability limitations of the present model. They are:*

Assumptions

1. The node delay of .001 seconds is constant for each message--based on Dr. Kleinrock's (UCLA) observations of the ARPA Network.

2. Node-traffic capacities are infinite.

3. Message buffers are infinite--based on Dr. Frank's (NAC) contention that this holds for lines with 80% or less utilization.

4. Poisson statistics apply to interarrival times and message lengths.

5. Arrival statistics are independent of message lengths.

6. There are no acknowledgement messages (to be relaxed).

7. There is no division of messages into packets (to be relaxed).

Limitations

1. All switching is store and forward (to be generalized).

2. Nodes are connected by full-duplex lines (to be generalized to allow for simplex lines.)

3. Channel capacities are assigned on the basis of a fixed sum of capacities rather than by cost (to change and allow for cost).

4. Fixed-minimum-linked routing is used (this has been shown to be nearly optimal by NAC).

5. Computer throughput is designated by a single number, the number of megabits (Mb) modified/sec.

6. All transactions are of equal priority (to be modified).

The ARPA Network as shown in Figure 2-1 was explored with the prototype model. The specific objectives being explored were a decrease in

---

*The comments in parentheses refer to planned near-term changes to the model.

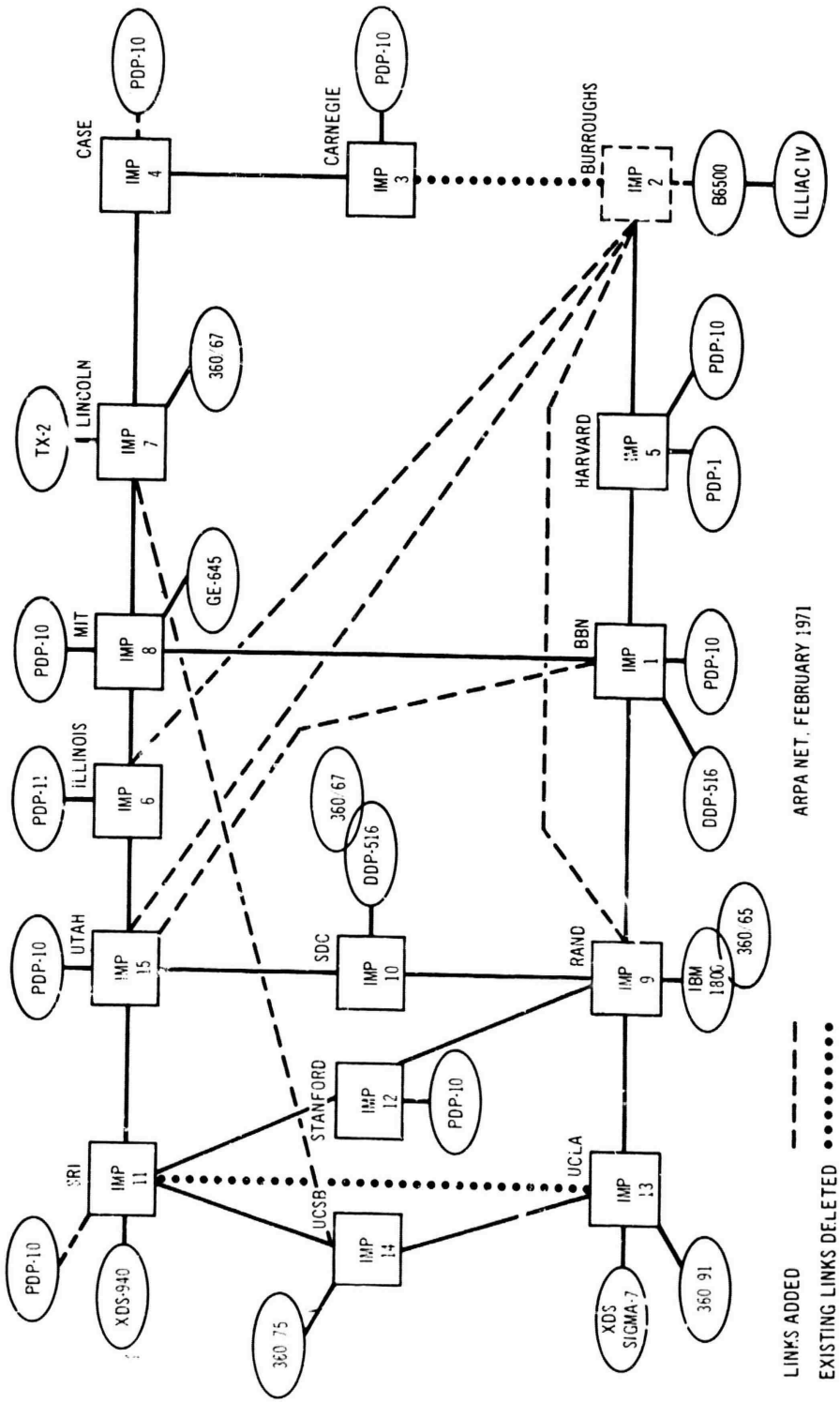


Figure 2-1. ARPA Network (with modifications)

i \ j														
	63	0	1	1	1	0	0	1	0	1	2	0	0	0	0
	0	18	0	3	0	3	0	0	0	0	0	0	0	0	1
	1	1	99	2	0	0	0	1	2	0	1	1	0	0	2
	1	3	2	63	0	1	0	0	0	0	0	0	0	0	0
	1	1	0	0	63	0	0	2	1	0	1	0	0	1	0
	0	25	0	1	0	38	1	1	0	0	1	0	0	1	8
	0	4	0	0	1	1	135	2	1	0	1	0	0	0	5
	1	2	1	0	1	0	1	90	0	1	1	1	0	0	1
	0	1	1	0	1	0	1	1	72	1	1	0	0	0	1
	0	1	0	0	0	0	0	1	1	45	1	0	1	0	0
	1	2	2	0	1	0	1	1	0	1	108	1	1	0	1
	0	2	1	0	0	0	0	2	0	0	2	63	0	0	0
	0	1	1	0	0	0	1	1	1	1	1	0	81	1	1
	0	8	0	0	0	2	0	1	1	0	1	0	1	171	5
	1	17	1	0	1	3	2	1	1	0	1	0	0	1	261

(a) Job Arrival Matrix (Number of Jobs per Day Sent from Node i to be Processed at Node j)

Node	Throughput	Node	Throughput
1	23	9	28
2	8312	10	33
3	14	11	21
4	14	12	14
5	16	13	268
6	7	14	44
7	39	15	14
8	35		

(b) Computer Processing Power (Throughput) at Each of 15 Nodes (in modified Mb/sec.)

Figure 2-2. Data Input for ARPA Network

bottlenecks, and a significant reduction in response time, by decreasing the average message path length traveled. Input data consisted of the job-arrival matrix in Figure 2-2(a) based on "congenial" node traffic (the (i,j)th entry is the number of jobs sent from node i to node j), mean message sizes of 70 kb (set large to make up for the low job-arrival rates and the lack of packets and acknowledgements); computing center throughputs shown in Figure 2-2(b), mean job sizes to be processed at node j of 15 x P_j where P_j is the throughput (modified megabits/sec.) of the computers at node j; and real mileage distances betwen nodes. All links were preset to 50 kb.

The results indicated that by adding six links (dashed lines) and deleting two existing links (dotted lines), the modified 15-node, 23-link network successfully accomplished the stated objectives. The mean communication response time was reduced from 57.4 seconds to 39.9 seconds. The average path length was reduced from 2.8 to 2.2 links traversed, and the diameter was reduced from 6 to 4. Bottlenecks were diminished from a worst case of 83 jobs/day (including 9 links of 50 or more) to a worst case of 40 jobs/day. The details of this experiment are currently being documented as a forthcoming SDC technical memorandum.

The model results have been discussed with Dr. Kleinrock of UCLA and Dr. Frank of NAC. Both felt that the model would be improved by breaking messages into packets and sending acknowledgements, and this will be done in the next period.

2.1.2 Plans

Improvements to the model during the next six months will include, in addition to the changes mentioned in parentheses above, the following capabilities:

1. Storage Capacity. For given traffic densities and processing capacities, the effects of limited storage capacity on network performance in general, and on alternate routing and load balancing in particular, will be evaluated. The impact on network efficiencies and costs of very large, very cheap, very fast memories (say, 10^{13} bits of laser or bubble memory, such as are promised by advanced technology) will also be a tradeoff of interest to CACTOS.

2. Error Rates. The effects of modifying error rates in computing and communications will be evaluated for various configurations and traffic densities.

3. Reliabilities. The effects of modifying reliabilities of network components will be evaluated for various configurations and traffic densities.

The tradeoff analysis will be based largely on the model results. Of particular interest during the next six months will be studies of centralized versus distributed networks, supernodes, and message versus circuit switching. It will also be possible to get graphic analysis of performance characteristics such as reliability, vulnerability, throughput, and security in a time and cost framework.

The Marine Corps Personnel Network system is being modeled with its existing configuration and its connections to AUTODIN. Snapshots of the system, reflecting the yearly growth in processing requirements over the years, will be taken to detect evolutionary changes in performance. Alternate configurations, including dedicated communication trunks, will be considered to determine their impact on network performance and costs. Distributed intelligence strategies for base-level computers may be considered.

The next step will be to construct an interface between the Marine Corps data base and the model. This will allow the user to access files for information on nodes. For convenience, this access will be called an executive; whether it will be on line or batch remains to be determined. The model's greatest utility is in an interactive mode, and on-line access would be desirable, but may require more development effort than can presently be given to it. The situation is being evaluated now.

Once the candidate networks are evaluated and the tradeoff experimentation is begun, the next step in model development will be to build a more general analysis model that can be used to analyze large, complex networks, such as command and control networks. The general model will have an executive that may access not only the data base, but modules containing (for example) simulation and statistical routines. These additional modules will not be identified until the prototype model is complete and some experience has been gained from the initial network analyses.

2.2 DATA BASE DEVELOPMENT PROJECT

2.2.1 Progress

The initial tasks in data base development have been to isolate the ingredients that would identify the central parameters, or characteristics, of a network and to determine what data could be made available from candidate networks. The results of these tasks are summarized below.

A portion of the data base is devoted to the network configuration, and is limited to the basic characteristics of nodes, links, and traffic; these basic characteristics will be adapted and enhanced as a result of the model analysis of the candidate networks. Data input included here is Network Identification, Node Record Indicator, Number of Nodes, Node Number,

Node Name, Node Location (Latitude, Longitude), Processor Type, Processor Number, Link Record Indicator, Number of Links, Source Node Number, Destination Node Number, and Line Number.

Eventually, with the general model, the data loaded will be either specified directly to the model as job and traffic matrices or loaded into the data base as a set of specifications that can be selected and applied to several network configurations. The first option is available to the user through the prototype model.

Finding a candidate DoD network was more difficult than first anticipated. Logistics and command and control networks were considered, but posed the drawbacks of being either classified or in procurement. After much effort, a candidate was found: The Marine Corps Personnel Network, which comprises major computers in Kansas City and Washington, D. C., and satellite computers in Vietnam, Hawaii, Camp Pendleton, Camp Lejeune, and Okinawa. It is dynamic, in part because of the scaling down of the Marine Corps effort in the Far East, and analysis over time is therefore possible. The satellite computers process command and control information as well as personnel data. All computers are linked through AUTODIN. Initial statistical data on messages, hardware, and AUTODIN have been collected from the Marine Corps, and the model network is now in the process of being run.

Although not directly part of the data base effort, the analysis of future technology will relate to it and to the modeling of future configurations and properties of networks. A series of white papers is being produced describing various developments. One, dealing with memory organizations and addressing, will soon be published; others are being written on communications, terminals, peripherals, and other areas related to networks. These will be published during the next reporting period.

2.2.2 Plans

In examining the future of the data base development, the model development must be kept in view. In order to model complex, large-scale networks, we will need a data base that can be accessed either in batch or on line by the model executive. The data base currently envisioned will contain data from Auerbach's Computer Characteristics Digest and Data Communication Reports, as well as data drawn from the ARPA Network, AUTODIN, and the Marine Corps network. Ultimately, it would be desirable to allow the user to specify the computer, peripherals, and terminals for each node by entering a simple code command. This would not only facilitate the initial modeling, but be valuable for on-line modification of the network.

The following milestones are expected to be achieved during the next six months:

1. Modeling of the Marine Corps Personnel Network.

2. Completion of tradeoff analysis.

3. Completion of technology forecasts.

4. Initiation of general data base and model development.

Beyond these milestones would be the modeling of larger networks, analysis of alternative futures, the possible incorporation of statistical and simulation tools for the analysis of the data base, and interactive data base (and model) access through the ARPA Network.

2.3 STAFF

Dr. B. P. Lientz, Principal Investigator

Model Development Project

Dr. R. L. Citrenbaum, Head
G. M. Cady
D. R. Lashier

Data Base Development Project

Dr. N. E. Willmorth, Head
L. G. Chesler
D. M. Gunn (part time)
R. Mosier (part time)

2.4 DOCUMENTATION

Citrenbaum, Ronald L. Analysis of Computation and Communication Model. SDC document SP-3601. In publication.

Gunn, Donald M. Survey of Digital Data Communications. SDC document SP-3603. In publication.

Mosier, Robert. Memories: Addressing and Organization. SDC document SP-3602. In Publication.

3. NATURAL COMPUTER INPUT/OUTPUT

As the power of both computer hardware and computer software increases and the price decreases, the computer becomes more omnipresent as both a tool and a necessity in our daily lives. As the expanding circle of contact between men and computers continues to increase, the disparity between man and computer in communications capability becomes more evident and less tolerable. Therefore, it behooves us to divert some of the available power of computer systems to mediate the communication gap and provide computer input and output systems that are "natural" and acceptable to men. Toward this end, the Natural Computer Input/Output task is providing research and development in computer processing and semantic interpretation of natural English, hand-drawn pictorial and symbolic input, computer-generated images, speech understanding by the computer, and computer-synthesized speech.

The Natural Computer Input/Output work is divided into three projects: the CONVERSE project (the development of an English data management system), the Graphic Input/Output project (recognition and utilization of hand-drawn and hand-printed input), and the Voice Input/Output project (speech understanding and synthesis). The interdependence between the Voice Input/Output and CONVERSE projects is both obvious and natural, and (though the two projects are at different levels of attainment) communication, cooperation, and commonality of basic intent are uppermost in their direction. The Graphic Input/Output project's primary concerns are with information whose content cannot be readily conveyed by either the spoken or the written word, but rather through pictorial or notational conventions best portrayed and conveyed in two dimensions. The major emphasis of this work has been on hand-printed input and computer-generated output of mathematics, developing applications requiring mathematical notation, and extending the notational capability into other domains.

Taken as a whole, the Natural Computer Input/Output task is providing the technological basis for operational man-machine systems for which the ultimate end user will require little, if any, special training in computer science.

3.1 CONVERSE: AN ENGLISH DATA MANAGEMENT SYSTEM

The principle goal of this project is to develop promising new natural-language processing techniques and to implement an experimental, prototype computer program system, based on those techniques, that will permit users to communicate with large on line data bases in ordinary English.

The two key objectives of the prototype system are (1) the ability to recognize a substantial subset of English and (2) the ability to store and search large quantities of conceptual and factual information. Particular emphasis is being placed on versatility--on developing a system that is potentially

applicable to the management of a wide variety of files, including large formatted files, files of complex relational data, and files of documentary information. A third objective is the development and demonstration of language processing and data retrieval techniques that are more powerful than previously available techniques, or more efficient, or both.

At the start of this contract year, we had: (1) implemented first versions of a natural-language compiler that recognized English surface syntactic structures and produced file-searching procedures in a formal intermediate language; (2) constructed a data management system that accepted these procedures and carried out specified storage, search, and computational operations; (3) developed a promising new approach to syntactic recognition in which all appropriate deep and surface structures are simultaneously produced (deep-structure representations generalize the syntax-analysis component and provide canonical trees upon which a simplified set of semantic rules can operate to provide intermediate-language procedures); (4) begun the programming necessary to extend the natural-language compiler to produce deep structures; (5) begun exercising the data management system with an initial data base of 4,000 facts; (6) implemented functions to create an initial concept network of semantic information used for disambiguation during input-sentence parsing and intermediate-language generation; and (7) made a start, in collaboration with Dr. L. Travis at the University of Wisconsin, on a promising new approach to large-scale data base inference-making.

The single impediment to further progress at the start of this reporting period was the severe restriction of available core-memory space in the LISP programming system used by CONVERSE on the IBM/360, despite our efforts to place more and more information on disc in an efficiently retrievable form. (At the present time all dictionary, concept-net, and fact-file data, and most grammar-rule information, are stored on disc.) Progress on both the CONVERSE system and the LISP system has been realized. We report on CONVERSE here and on LISP in section 5.3.

3.1.1 Progress

The current status of CONVERSE-360 is described in a paper presented at the April, 1971, ACM Symposium on Information Storage and Retrieval. This paper has been published in the symposium proceedings.

During this past quarter, two important milestones were achieved: (1) the demonstration of a limited-core-memory version of CONVERSE (version V-Ø) and (2) the realization of a new natural-language compiler in the expanded version of the LISP 1.5 system. Other key areas in which progress has been made include data management and intermediate language, deductive inference, and the analysis of fundamental syntactic and semantic relations.

CONVERSE V-Ø

Despite severe core-memory restrictions, an initial demonstrable system was achieved at the midpoint of this reporting period. The system is implemented in two 46-page copies of LISP 1.5 running under ADEPT/67. The system translates questions of moderate complexity into intermediate language using only surface-structure information and data from a small concept network. Questions are then answered by searching our initial data base. A sample V-Ø printout is shown in Figure 3-1.

In pushing this demonstration milestone to a successful completion, we have solved a number of minor but time-consuming problems in intercommunication between the two LISP programs and between the programs and their associated disc data files. All of the checked-out facilities in V-Ø will be of direct use in future, larger-scale versions of CONVERSE (versions V-1/V-2).

85-Page LISP Natural-Language Compiler

The second major milestone was achieved late in the reporting period when staff members completed an extensive series of revisions and improvements to our LISP programming system. The most important result of this effort was the ability to construct 85-page LISP programs. (This work is reported separately in section 5.3.)

As soon as the new large LISP became available, we implemented a new natural-language compiler with facilities for producing both surface and deep structures. As the reporting period came to a close, we were producing a number of correct surface and deep structures and were well along in the process of writing and implementing rules to produce intermediate language from deep structures.

In syntax, progress has been made in both extending the scope of the parser's input and improving the quality of the parser's output. The range of English structures that can be given a surface-structure parsing has increased to include nominalizations and sentential complementation (of the FOR-TO, THAT, and POSS-ING varieties), superlatives, tag questions, and verb particle constructions. By revising the Structure Building (SB) rules, we have continued to eliminate undesired surface parsings for any particular string.

Revising the SB rules has also improved the quality of the deep structures output. This end has also been achieved by (1) debugging the operation of the Structure Changing (SC) rules (which perform such basic operations as adjunction, deletion, and replacement on trees) and (2) adding new SC rules that reconstruct elements deleted in the surface string from propositions semantically inherent in that string. The latter effort naturally results in greater success in representing paraphrase relations. Special efforts have been made to represent the wide variety of English comparative structures in such a way that they are amenable to a relatively simple semantic treatment.

```
 WHAT IS THE POPULATION OF GARDEN GROVE?
(((((POP*60)) 84238)))

-


 WHAT CITIES HAVE A POPULATION THAT IS GREATER THAN GARDEN GROVE'S?
(((OBJECT (ANAHEIM)
   (BERKELEY)
   (BURBANK)
   (FRESNO)
   (GLENDALE)
   (LONG BEACH)
   (LOS ANGELES))))

-


 WHAT IS THE POPULATION FOR ALL SMOGGY CITIES?
(((OBJECT (BEVERLY HILLS)
   (BURBANK)
   (DOWNEY)
   (GARDENA)
   (LONG BEACH)
   (LOS ANGELES))
 (((POP*60)) 30817 90155 82505 35943 344168 2479015)))
```

Figure 3-1. Question Answering in CONVERSE V-Ø

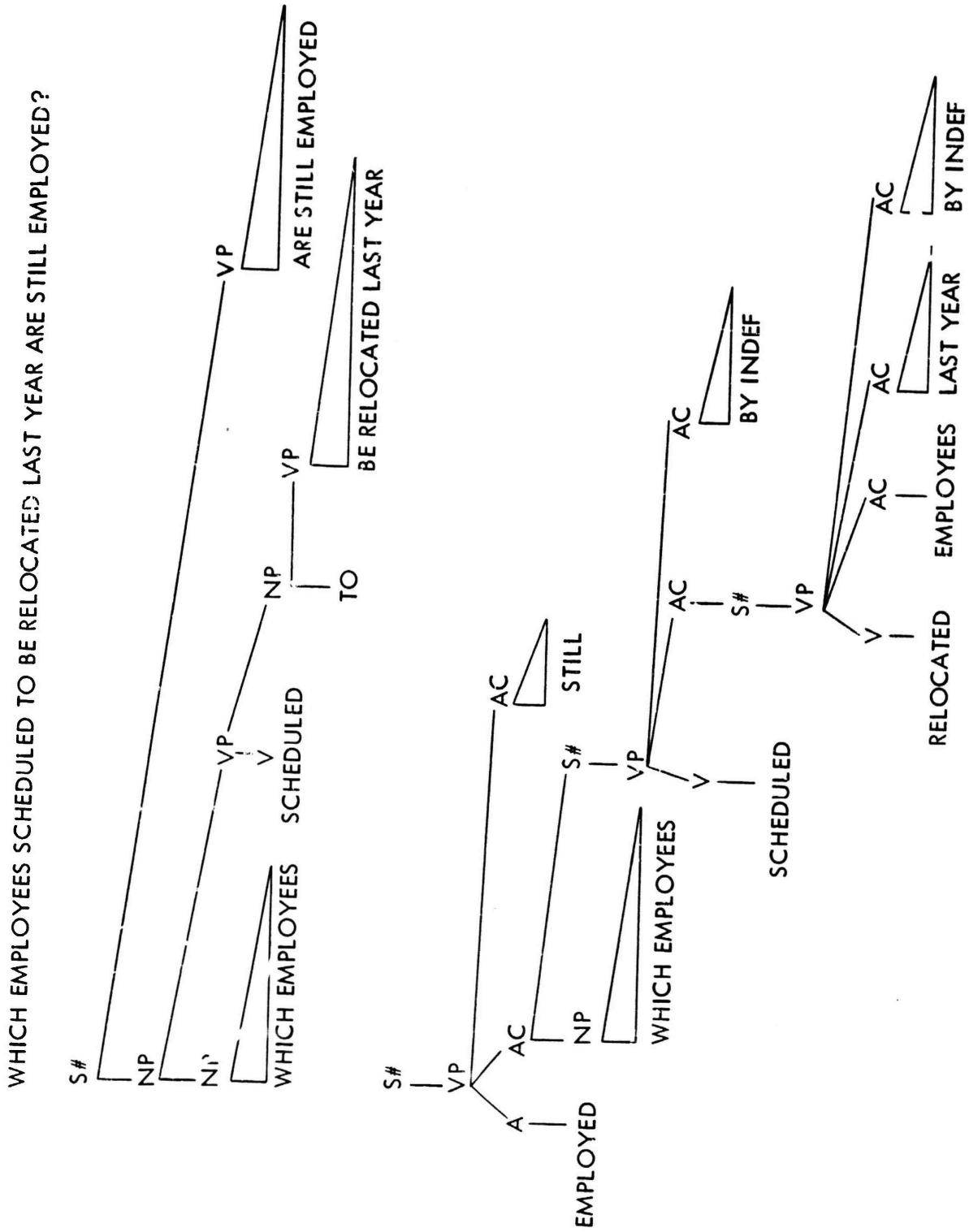


Figure 3-2. Surface (top) and Deep (bottom) Structures Produced by the Natural Language Compiler

Figure 3-2 illustrates at least two points about deep structure: First, the deep structure contains in explicit form information that is only implied in the surface structure; second, the deep structure has a canonical form for each proposition.

To achieve an interface between syntax and semantics, i.e., intermediate language, a set of Semantic Interpretation (SI) rules were developed. These rules take as input deep-structure trees and produce as output procedures in intermediate language. The SI rules are treated as a separate component to facilitate revision and debugging of the syntax and SI rules and to accelerate run time by avoiding unnecessary semantic work on syntactic structures that are aborted in the parsing process.

In the future, the semantic rules will incorporate case assignment. The case framework employed by CONVERSE provides a strategy for mapping each distinct function of a prepositional phrase and noun phrase into the relevant semantic categories in the data base. Restricting possible case relations to those that might be encountered in a practical data base has greatly simplified the choice of possible cases.

The preliminary task of extracting a relatively comprehensive set of distinct classes of prepositional phrases is now completed. These classes form the basis of a first list of cases. Effort in this direction is continuing in two steps: (1) examination of the syntactic and semantic information associated with each prepositional phrase for clues concerning its case membership; and (2) the utilization of this information to write case-assignment rules.

Intermediate Language and Data Management

We are presently working out a number of operators to be added to the formal intermediate language (IL). These changes will make IL easier to generate from deep structures and will increase its expressive power. For example, we have developed operators to handle quantification (e.g., "Every girl was kissed by some boy."), compound negation (e.g., "No Pole knows a German whom he does not like."), and complex comparisons among relations (e.g., "Which flights depart from midwestern cities for cities further East?").

A new, 85-page version of the CONVERSE data management system has been produced, and a second data base of more than 10,000 facts has been constructed for more extensive exercising of future versions of CONVERSE.

Inference-making

Work on an inferential system for CONVERSE during this reporting period has focused on developing specifications for the deduction grapher, which is the

part of the system that draws inferences from general facts. Programming of one part of the deduction grapher has commenced, and the design of a driver language is underway. This driver language is the CONVERSE intermediate language extended to enable its use for specifying deduction-grapher strategies and operand expressions.

The deduction grapher is designed to make inferences in a question-answering system rather than in a formal mathematical system. The difference is that, in a question-answering system, an essential (and perhaps the most difficult) part of successful inference is the selection of relevant premises from a very large set of premises (most of which are irrelevant to the inference being attempted), while in a formal mathematical system, premise selection does not present a problem. On the other hand, showing in a question-answering system that selected premises have needed deductive relationships to each other does not present the problem it does in mathematical inference because the relationships are likely to be simple ones among many different predicates.

The deduction grapher proceeds by generating proof proposals, progressively filling out the details of these proposals, and filtering out bad proposals at various stages along the way. Bad proposals are those that cannot be filled out to become valid proofs. If all originally generated proposals are filtered out, the system loops back to generate additional ones.

General facts are stored in an associative network called the premise graph. This graph is composed of formalized assertions explicitly linked together at the points where they can possibly interact with each other deductively. As with mechanical theorem-provers based on the resolution principle, the assertions are in Skolemized, quantifier-free form, but the deduction grapher uses primitive conditionals as its normal form rather than the conjunctive normal form of resolution systems. The explicit, premise-connecting links in the premise graph represent Prawitz-Robinson unifications of literals, i.e., of the atomic components of premises. In contrast to previous automatic theorem provers, in the deduction grapher first-order unifications (i.e., unifications between literals in "original clauses") are discovered when a general fact (a premise) is entered into the system, and they do not need to be recomputed at each inference attempt. Rather, they become a permanent, defining part of the premise graph, making it possible for the deduction grapher to discover very quickly premises appropriately interconnected for its deduction tasks.

Since the premise graph can potentially get very large, the first thing that the deduction grapher does when given a deduction task is to discover plausible paths to middle terms. These middle terms serve to identify likely nodes in the premise graph from which proof proposals can be developed. Several techniques for generating middle-term paths are being investigated. The one that currently looks most promising is the iterative multiplication of a predicate connection matrix by itself.

The subgraphs picked out of the premise graph as proof proposals represent proofs that would be valid considering only truth-functional structure. They serve as useful plans for working out complete proofs, i.e., proofs that are also valid from the standpoint of quantificational structure. The approach used to validate proof proposals is to discover possible collisions of substitutions for the variables contained in the premises used in a proof proposal, and then to test whether the possible collisions are actual. To be a valid proof, a proposal must be collision free.

An important component of the deduction grapher is the truth evaluator (T-E) described in the final report for the previous contract period.* Its importance lies in the fact that most of the information used in question-answering inference is concrete (logically of the form "P(a)" or "R(a,b)" where "a" and "b" are not variables but proper names). There are so many concrete facts that they cannot be efficiently stored in the premise graph; instead, they are stored as data sets in the CONVERSE fact file. The T-E component is called when needed by the deduction grapher to retrieve information from the fact file.

Analysis of Fundamental Syntactic/Semantic Relations

Two approaches have been followed in developing an affixal component for CONVERSE. In the first approach, about 10,000 definitions of suffix-words were extracted from the machine-readable transcript of Webster's Seventh New Collegiate Dictionary (W7) and sorted according to the suffix and part-of-speech shift by which they had been formed. One hundred and fifteen definitions were obtained for adjectives formed by adding '-ful' to nouns, and 900 definitions for nouns formed by adding '-ion' to verbs. Alphabetization of the definitions within each group usually brings out quite clearly the major semantic functions of each suffix and often suggests semantic categories in terms of which selectional constraints on the suffix can be stated. These constraints will enable CONVERSE to predict, in favorable cases, which semantic function is appropriate for a given use of the suffix.

In the second approach, we have sought to select and specify notions suitable for inclusion in CONVERSE's concept network so that the semantic functions of each of the more frequently used suffixes can be applied by CONVERSE to derive the meanings of words formed by every such suffix from the meanings of their base forms. Keywords of the defining formulas used in W7 definitions of suffix-words, when taken in the senses in which they are used in

---

*Weissman, C. Computer-Aided Command: Final Semiannual Technical Summary Report to the Director, Advanced Research Projects Agency, for the period 16 March 1970 to 15 September 1970. SDC document TM-3628. September 1970.

those formulas, provide an initial set of affixal relations suitable for this purpose. Conceptual analyses previously prepared for this set of relations are being refined in the light of comprehensive data now being obtained regarding coordination patterns among the defining formulas in W7. As soon as this refinement has been completed, we will prepare brief conceptual analyses of the notions underlying the case relations now recognized by the CONVERSE grammar. It is already apparent that there will be considerable overlap not only between these two sets of relations but also between each and a set of 60 notions involved in thematic relations for which conceptual analyses have already been prepared. The notions falling within the intersection of these three sets are the ones that we will enter first in CONVERSE's concept network. We anticipate that these additions to the network will not only provide CONVERSE with a significant capability for interpreting affixes, syntactic relations, and thematic relations, but also facilitate the statement of general facts for use by other components of CONVERSE.

Cooperation with Other Projects

We are continuing to collaborate closely with the Voice Input/Output project in the long-range objective of attaining a vocal CONVERSE. We believe that present efforts at producing deep structures and the forthcoming capabilities for inference making will be especially useful in reaching this long-range objective.

3.1.2 Plans

Plans for the next six months center on two demonstration milestones: CONVERSE V-1 and V-2. V-1 will be demonstrated during the next quarter. This version will demonstrate a considerably enhanced question-answering capability, one that utilizes our new deep-structure parser and a richer data base. The second milestone, CONVERSE V-2, to be reached in the last quarter of the contract period, adds to V-1 a capability for recognizing and interpreting declarative and imperative sentences.

We will demonstrate user feedback, and features that promote user confidence, in V-1. The user-extensible features of CONVERSE will be demonstrated in V-2. We will continue efforts towards the ultimate integration of the deductive grapher program and files into the overall CONVERSE system.

Finally, we will begin to explore network applications of CONVERSE and the question-answering systems of other ARPA nodes.

3.2 GRAPHIC INPUT/OUTPUT

Enhancing communication between man and computer by providing functions and facilities that are compatible with, or equivalent to, techniques used in visual man-to-man communication is the principal goal of the Graphic Input/Output project. Our concern has been to develop methods of graphical input and output that will permit a user to carry on a dialogue with a computer in the language and notation of his discipline or problem domain.

The functional entities required for such a dialogue are a data-input tablet (e.g., the RAND Tablet), an interactive CRT display operable as a terminal in a time-sharing system, and a character-recognition program that will accept the symbols used in the notational expressions.*

The near-term project goal is to develop programming systems that utilize two-dimensional notation. Our initial effort is to use mathematics, the most ubquitous scientific notation, for numeric and symbolic programming. The work is based upon previously developed programs that accept two-dimensional mathematical expressions hand-drawn on the input tablet, extract the explicit and implicit information from them, and transform them into representations amenable to existing processing techniques.

3.2.1 Progress

The process of converting our existing AN/FSQ-32 Time-Sharing System programs to the IBM/360 ADEPT Time-Sharing System has continued during this period. All programs except the Unparser, the program that converts a linear representation of a mathematical expression into a high-quality two-dimensional representation, are now converted and operational.

In the process of conversion, a new dictionary-building program, one more efficient and convenient than any of its predecessors, was implemented for the character-recognition program.

The Adding Machine (TAM)

An initial language using mathematical notation, called TAM (The Adding Machine) has been designed and is being implemented. TAM allows arithmetic manipulation using a powerful set of operators as constants, variables, and one-dimensional or two-dimensional arrays. It provides looping facilities, single-statement functions, and user-defined input and output. Some built-in functions, such as square root and logarithm, and built-in constants, such as Π and e, are provided. TAM is an incremental system: each statement is executed before the next statement is requested.

---

*Bernstein, M. I. Hand-Printed Input for On-Line Systems. SDC document TM-3937. April 1968.

TAM is being implemented with SDC's compiler-writing systems. The interpreter has been designed and implemented, and is being debugged with dummy inputs from a tape file. The interconnection routines between the parser--which reduces two-dimensional mathematical notation to one-dimensional strings--and the interpreter have been written and are being debugged.

Dictionary Building

To build a dictionary, the user must begin by supplying samples of his printing. In Figure 3-3(a) the program has requested samples and the user has responded. In Figure 3-3(b) the input characters have been supplied to the recognizer and have not been recognized, as indicated by the '?'. There is one '?' for each unrecognized input stroke, aligned with the stroke, so that the user can easily determine which characters were not recognized. The user may now select the alphabet, appearing at the top of the screen, used for defining the input characters. The choices, selected by the light button at the lower right corner, are S, special characters--mostly punctuation and mathematical symbols; G, Greek letters; N, numbers--the alphabet shown; U, upper case; and L, lower case.

The user defines a character by encircling it and touching the appropriate character in the alphabet at the top of the screen, as shown in Figure 3-3(c). The system responds by entering the definition in the dictionary and then again applying the character recognizer to the input string, with the result shown in Figure 3-3(d). More than one character can be defined at a time, as shown in Figures 3-3(e) and (f). Figures 3-3(g) and (h) show the definition of an alphabetic character.

The dictionary builder also contains a TEST mode that allows the user to test the dictionary he has built. The user provides hand-printed information, as shown in Figure 3-4(a). The input characters that are recognized are replaced by generated characters of the same size and position, as shown in Figure 3-4(b). At any time, the user may return to the dictionary-building mode and use the current input to add to the dictionary, as shown in Figures 3-4(c), and (d), and (e). He may then return to the TEST mode, Figure 3-4(f).

Implementation Activities

Work has continued on implementation of the character recognizer in the Honeywell DDP-516. Currently, the feature-extraction portion of the recognizer is operational; other portions, including dictionary lookup and a rudimentary dictionary-building routine, are coded but not tested.

We have designed a multiplexor to allow an additional data tablet, a Graf-Pen, to be connected to the DDP-516. Used with an ARDS display terminal, it will provide another graphic I/O console. We have written software to

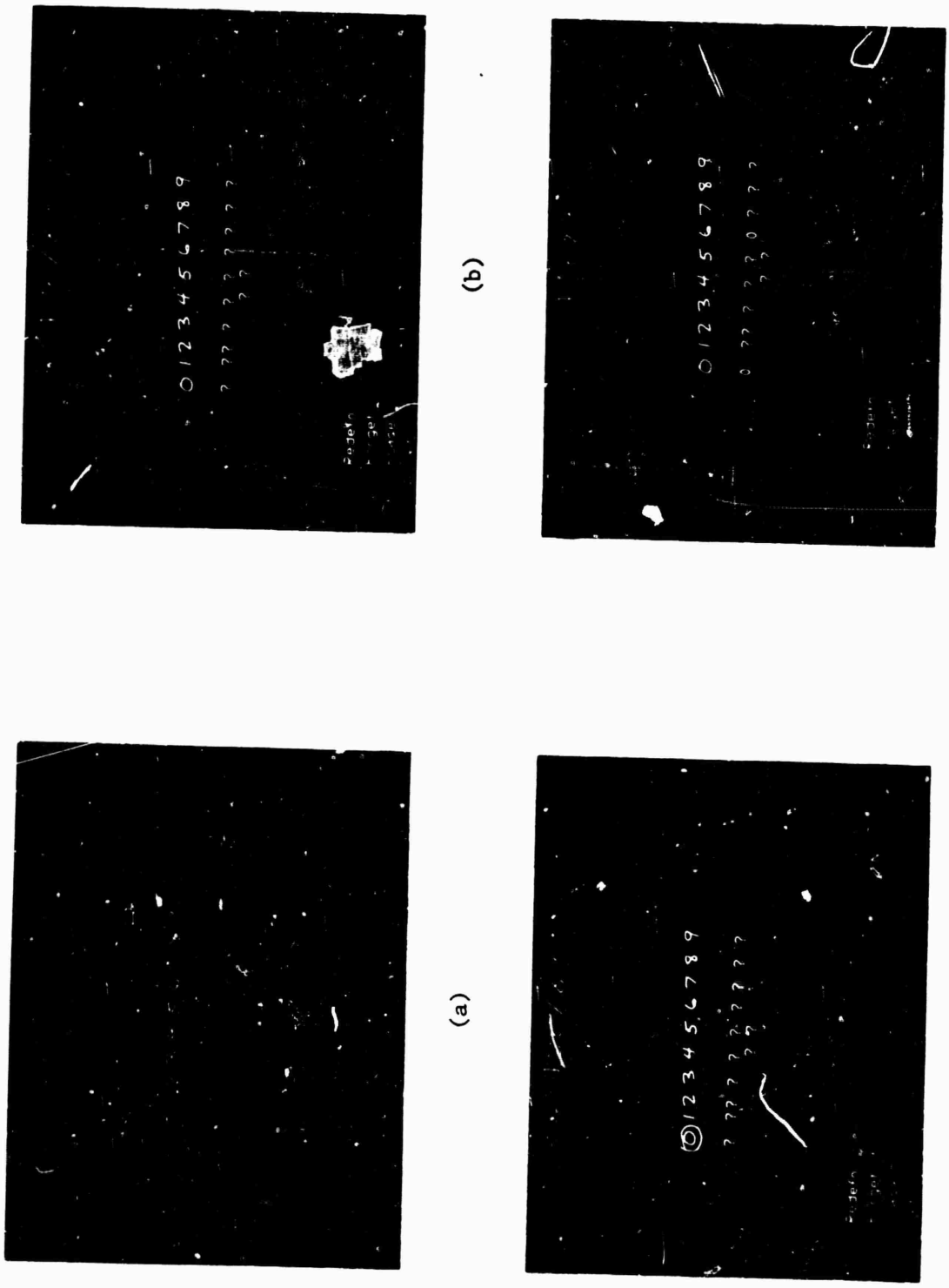
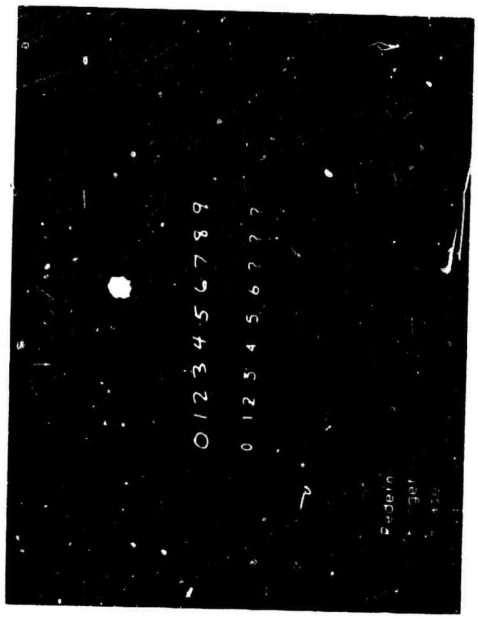
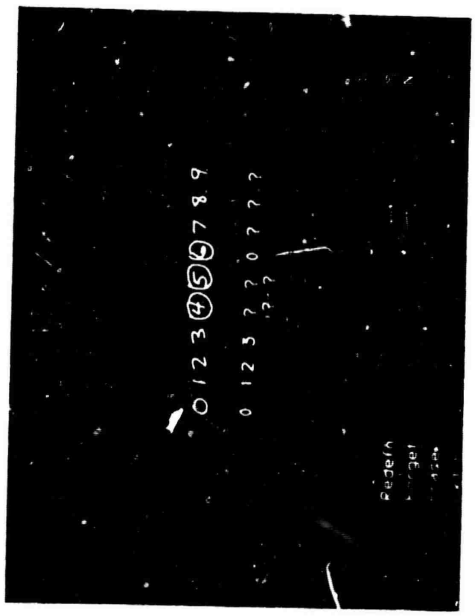


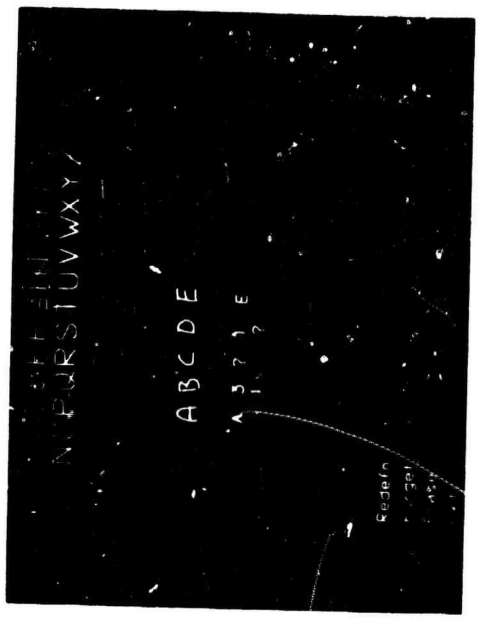
Figure 3-3. Dictionary Building



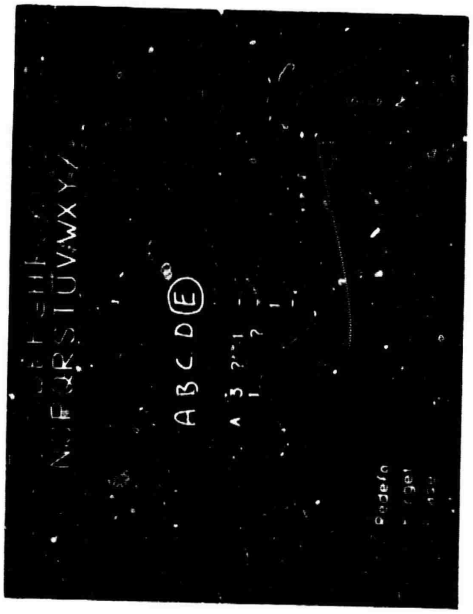
(f)



(e)



(h)



(g)

Figure 3-3. Dictionary Building (Cont'd)

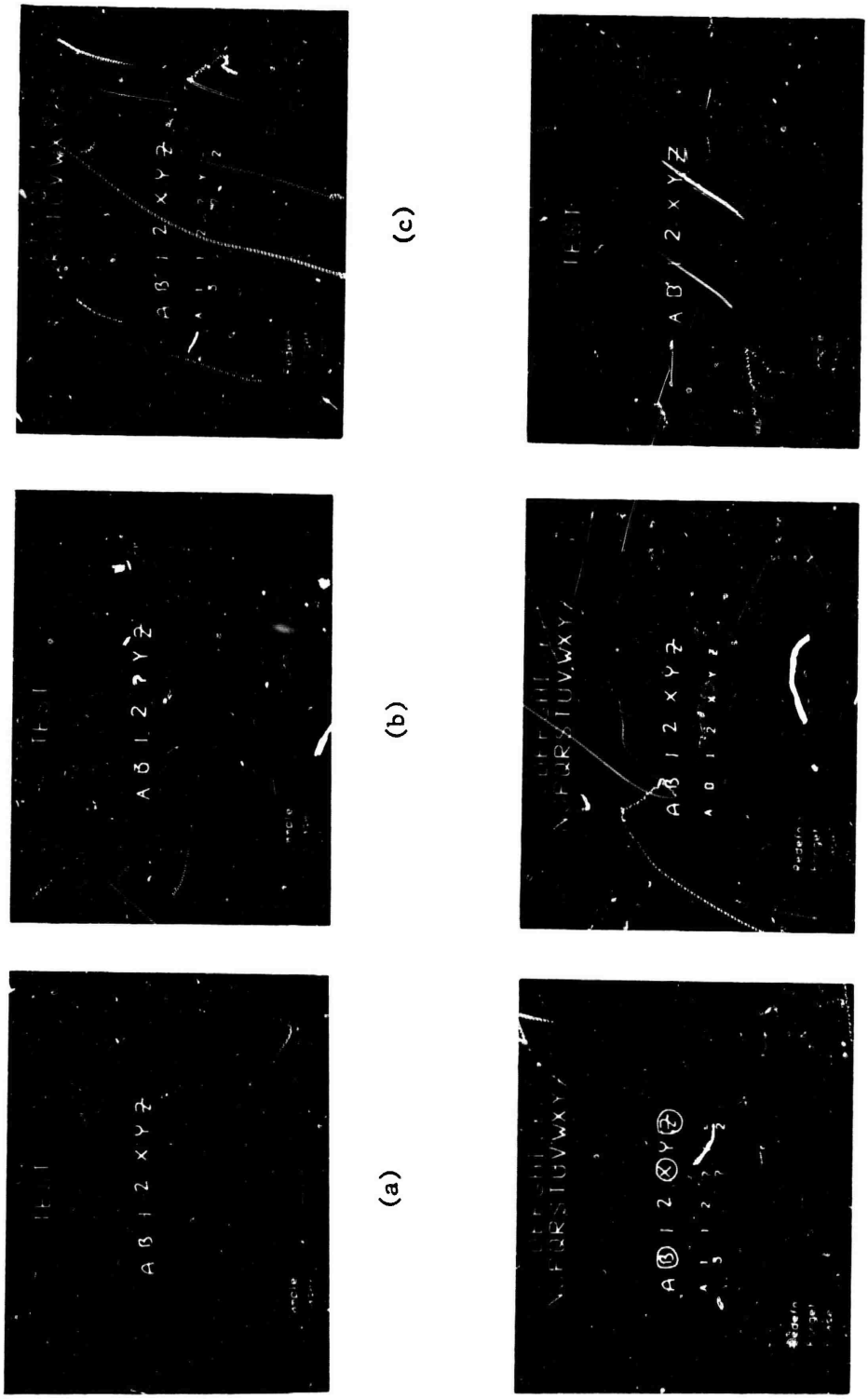


Figure 3-4. TEST Mode

convert a display buffer intended for the Beta Instruments display for use with the ARDS terminal. This allows console interchangeability.

3.2.2 Plans

We expect to complete a demonstrable version of TAM by the end of the contract period. This version will include the capabilities of the Unparser to display user-defined functions. As TAM is completed, we will begin work on a symbolic processing system for mathematics, providing operations such as algebraic manipulation or symbolic differentiation.

We expect to finish our implementation of a character-recognition program in the DDP-516 minicomputer. This should provide faster response for the user by lightening the processing load in the IBM 360 and will further isolate us from changes in the parent computer hardware. We also plan to implement, within the character recognizer, recognition of block-diagram symbols--blocks, circles, and lines. This will enable us to expand our use of two-dimensional notation and flow charts and block diagrams.

We have been studying means for making our graphics programs--in particular, the character recognizer--available on the ARPA Network. We will participate in the development of network graphic protocols to ensure that data-tablet and other information can be accommodated.

3.3 VOICE INPUT/OUTPUT

The long-term goal of the Voice Input/Output project is the operation of SDC's CONVERSE system with vocal speech input and output. CONVERSE (see section 3.1) is a natural-language question-answering system that employs a user-extensible subset of English and that aims, eventually, at employing a virtually unconstrained subset of English for data base management. In order to achieve a vocal CONVERSE, we will have to attack and solve almost all of the outstanding research problems regarding the recognition of speech by a computer--including the recognition of continuous speech (as opposed to the recognition of single words), the ability to handle large vocabularies, and the development of scanning processes and system integration techniques that allow parsing and disambiguation of noisy input. Because of its difficulties, the task of solving these problems must be approached in increments. The first two increments, a reimplementation of the Vicens-Reddy speech recognition system from the Stanford University PDP-10 to the SDC IBM 360/67 and the implementation of a vocal data management system that employs a highly constrained subset of spoken English, are the immediate goals toward which the Voice I/O project is working during the current contract year.

3.3.1 Progress

During the past six months, project efforts have focused on three major objectives:

1. The implementation of a Voice Laboratory, including the competitive procurement of a computer to support the project's activities.

2. Complete reimplementation of the Vicens-Reddy speech recognition system from the PDP-10 to the IBM 360/67, and the development of software necessary to convert the system as implemented on the IBM 360 for operation on the project's own computer.

3. The development of procedures for acoustic processing and system integration.

Progress toward these objectives has been substantial. A competitive procurement held among nine computer manufacturers resulted in the selection of a Raytheon 704 minicomputer. The physical facilities for a Voice Laboratory are now available to house the computer and a sound booth. The reimplementation of the Vicens-Reddy speech recognition system on the IBM 360/67 has been completed, and additional speech-analysis subsystems have been programmed and added to the Vicens-Reddy system to expand its capabilities. Several systems procedures for processing continuous speech have been developed and are being documented.

Voice Laboratory

The Voice Laboratory (see Figures 3-5 and 3-6) is now complete, and, with the delivery of the Raytheon 704 computer on 1 February 1971, work is progressing on a completely integrated speech-data-processing facility. The Raytheon 704 combines excellent signal processing capabilities with better than normal software for a minicomputer.

As has been noted in previous reports, the Voice Laboratory uses highly controlled acoustic and audio systems (now in conjunction with the Raytheon 704). The controlled audio system will assist in speech-data input, output, preprocessing, and analysis, and will provide on-line and magnetic-tape interfaces to other computers and, possibly, the ARPA Network. A key physical facility within the laboratory is a small, broadcast-quality sound booth designed to allow interactive use of a quiet display terminal and a microphone. Speech is collected by a high-quality condenser microphone and amplified by broadcast audio equipment. A patch panel, a tape recorder, and monitoring equipment allow a wide variety of speech-data collection, interactive speech experimentation, and voice-output activities to take place within one facility.

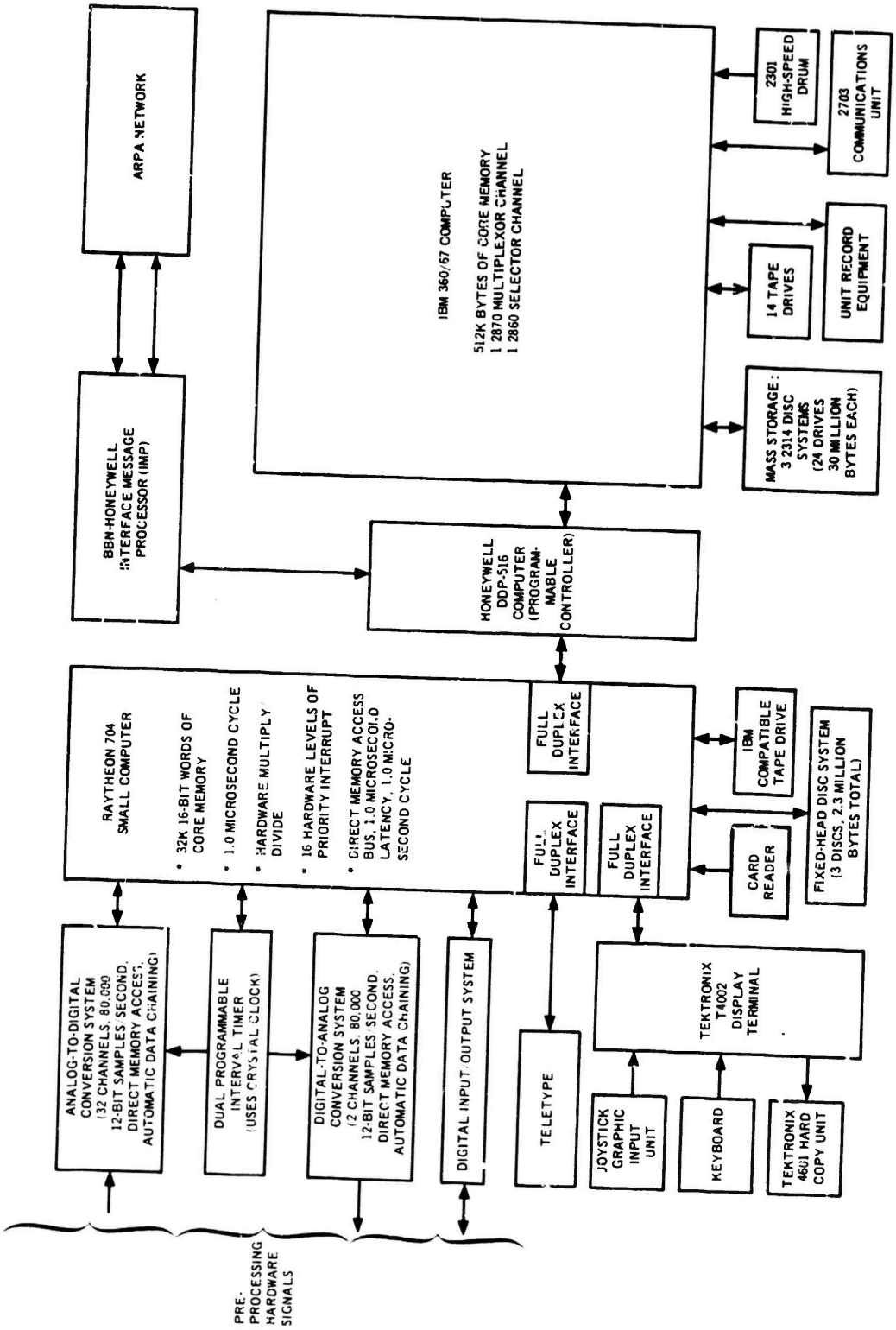


Figure 3-5. Voice I/O Project Computer Configuration and Other Computing Resources

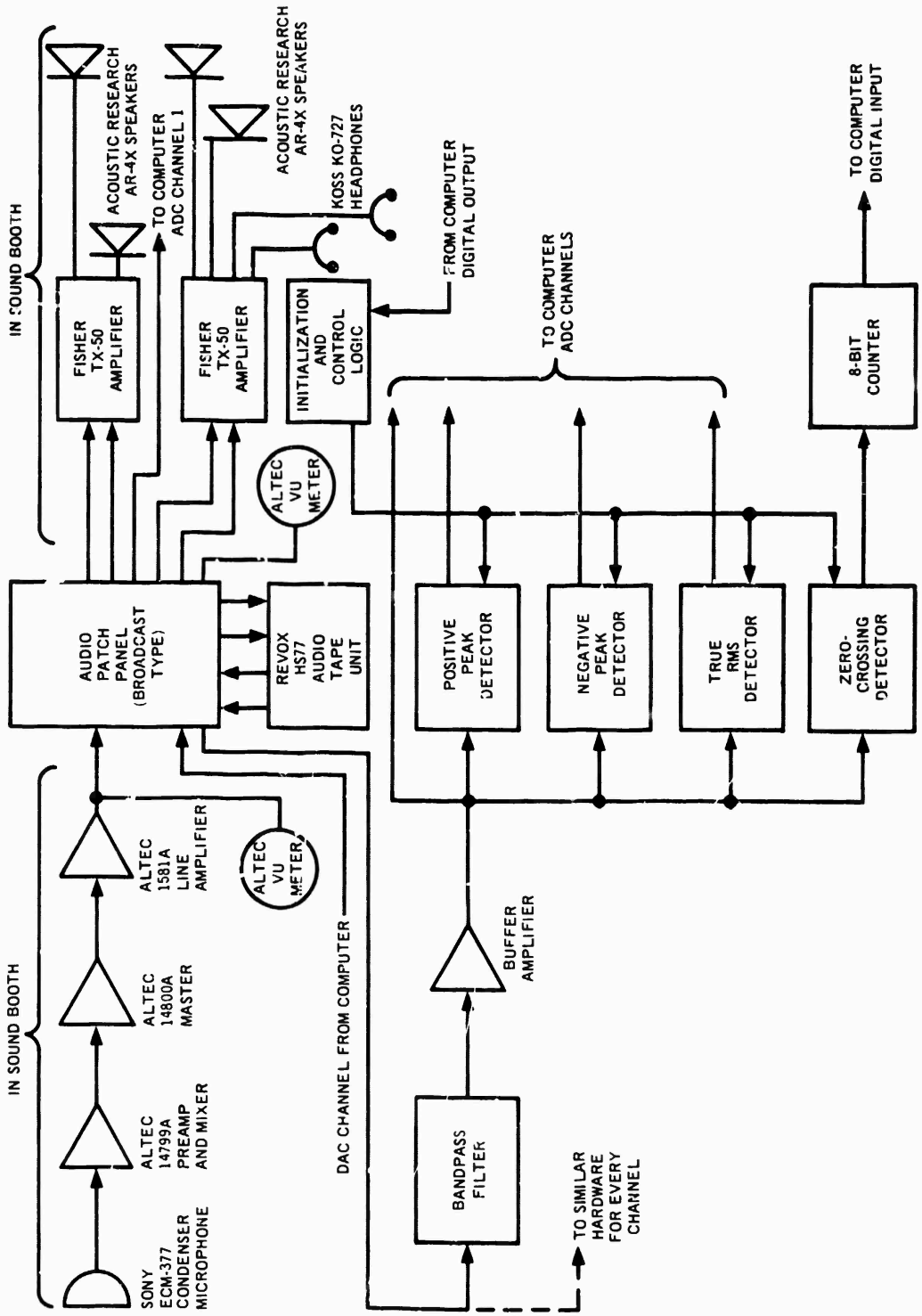


Figure 3-6. Input Hardware for the Voice I/O Laboratory

Software Development

During the past six months, the reimplementation of the Vicens-Reddy speech recognition system on the IBM 360/67 has progressed to the point at which coding is complete and checkout is in progress. Four documents (Kameny and Ritea, TM-4652 series; see section 3.5) describe the preprocessing, segmentation, and recognition subsystems, and explain the heuristics and algorithms that have been used but not heretofore documented. The lexicon-building and retrieval package is coded and operating but not yet documented. A math library consisting of several versions of Discrete Fourier Transforms, Fast Fourier Transforms, Chirp-Z Transforms, and digital filters is now being tested for execution time, core requirements, and accuracy, and will be documented when testing is complete. The major remaining task is the conversion of the system from the IBM 360 to the Raytheon 704.

Several software packages are being developed for the Raytheon 704. A mini-LISP has been coded but is not yet checked out. Ultimately, it will allow several programs that are now operating on the IBM 360 to run on the 704 in combination with other analysis tasks. Data structures in the mini-LISP are character and variable-length identifiers, nodes, and small integers. (If necessary, floating-point numbers will be added at a later date.) We are also developing a graphics package, a line-printer simulator using the TEKTRONIX storage tube display and hard-copy unit, absolute loaders for disc and magnetic tape, various utility programs, and a processor that simulates parallel, breadth-first evaluation. The latter will free us from having to predetermine the order of evaluation of pattern rules, which, because they are not unique, may produce several different "correct" identifications of the same acoustic segment. Predetermining the order of their evaluation would necessitate eliminating some that might either be correct or provide useful additional information in cases of ambiguous identification (such as may occur with the consonants "b" and "t" in different words).

System Procedures

In order to attack the problems of acoustic processing and system integration, two parallel efforts are in progress. The first is the development of CWIPER, a subsystem to perform all acoustic-oriented chores. These include collecting the speech sample, performing initial segmentations, extracting features, and searching for items in the utterance. Parts of CWIPER are modeled on the Vicens-Reddy system; others include additional hardware feature-extraction procedures and a new mapping and recognition procedure that does not require the input utterances to be completely mapped into the symbolic pattern of the lexicon items.

The second effort involves the specification and development of the model interface sequencing procedures to construct the total system. The mini-LISP system is being built with modules that represent all parts of the system except those that reside in CWIPER. These modules will be simulated with symbolic, rather than acoustic, inputs. This approach is being taken to allow the testing of algorithms on data of known accuracy. Hopefully, the tested algorithms will be combined with the actual acoustic processors to make an initial version of the vocal data management system operational during the next six months.

3.3.2 Plans

During the next six months, work will continue in several areas. The Vicens-Reddy system will be checked out on the IBM 360 and converted to the Raytheon 704. Several modifications are scheduled to improve the segmentation and mapping processors; they include improved classification of fricatives using the true-RMS detectors and better handling and identification of transient segments. The mini-LISP system for the Raytheon 704 will be completed and integrated into the processor library. Hardware development of the acoustic and digital conversion subsystems will be completed. The sound booth will be tuned to achieve the optimal, flat response for which it was designed. MUSIC V, a FORTRAN program written by Max Mathews at Bell Labs, will be transferred to the Raytheon 704 to allow us to test our hypotheses about certain acoustic rules.

Completion of CWIPER and the interface model will make possible the construction of the continuous speech recognizer. The prototype vocal data management system will demonstrate the reliability and appropriateness of the techniques that have been developed.

3.4 STAFF

Natural Computer Input/Output Staff

M. I. Bernstein, Manager

CONVERSE Project

C. H. Kellogg, Principal Investigator
J. H. Burger
T. C. Diller
K. J. Fogt
J. C. Olney, L. Travis — Consultants (part time)

Graphic Input/Output Project

T. G. Williams, Principal Investigator
Joan Bebb (part time)
Jean Igawa
J. P. McGahey
Jean Saylor

Voice Input/Output Project

J. A. Barnett, Principal Investigator
C. R. Kalinowski
Iris Kameny (part time)
L. M. Molho
H. B. Ritea
R. DeCrescent (part time)

3.5 DOCUMENTATION

Kameny, Iris, and H. Barry Ritea. Analysis and Development of the Vicens-Reddy Speech Recognition System: Table of Contents for Document Series TM-4652. SDC document TM-4652/000/00. December 1970.

Kameny, Iris, and H. Barry Ritea. Introduction and Overview of the Vicens-Reddy Speech Recognition System. SDC document TM-4652/001/00. December 1970.

Kameny, Iris, and H. Barry Ritea. Description and Analysis of the Vicens-Reddy Preprocessing and Segmentation Algorithms. SDC document TM-4652/200/00. December 1970.

Kameny, Iris, and H. Barry Ritea. Description and Analysis of the Vicens-Reddy Recognition Algorithms. SDC document TM-4652/300/00. March 1971.

Kellogg, C. CONVERSE Plan (1/5/71 - 9/5/71). SDC document N-(L)-24445. February 1971.

Kellogg, C., J. Burger, T. Diller, and K. Fogt. "The CONVERSE Natural Language Data Management System: Current Status and Plans." Proceedings of the Symposium on Information Storage and Retrieval, April 1-2, 1971, University of Maryland. Pp. 33-46. Published by the Association for Computing Machinery.

Travis, L. A New Approach to Implementing and Using Inference in a Question Answering System: Plans for CONVERSE. SDC document N-24463. February 1971.

4. SYSTEMS RESEARCH

The systems research tasks involve the development of new hardware and software tools and techniques that are important to the scientific community and contribute to the fabrication of a computer-assisted planning system. These projects presently include the ARPA Network development and the Graph-Meta analysis work, with each effort including both design and implementation aspects of the overall system development. In the case of the Graph-Meta research, the design is based on a rigorous analytical foundation in graph theory, while the network efforts are more application oriented, involving the HOST-to-HOST protocol development and studies of distributed data-base systems in a computer network. Both projects have been quite active during the reporting period, and will soon have facilities available for experimental usage by the other projects.

4.1 NETWORKS

The goal of the Networks project is to make the SDC ADEPT Time-Sharing System an operating part of the ARPA Network, and to explore ways in which it can both contribute resources to the ARPA community and benefit from the services the community makes available to us. As part of our network effort, we are also investigating "the distributed data base problem", and have considered a number of possible ways of integrating dissimilar data management systems. Because of the needed long lead-time for developing a feasible, practical approach, this effort is being pursued in parallel with the HOST-to-HOST protocol implementation, so that we can utilize the ARPA Network for experiments when ADEPT is integrated into it.

The ADEPT system runs on an IBM 360/67 and utilizes a Honeywell DDP-516 peripheral computer as an interactive I/O handler. The ARPA Interface Message Processor (IMP) is also connected to the DDP-516 computer, which collects and passes messages between the Network and the IBM 360. The programs that accomplish this in the DDP-516 and in the IBM 360 have been coded and debugged and are in use now in our work to integrate ADEPT into the Network.

The subsystem of ADEPT that interfaces with the Network is called HOSTOSS (Host Operating SubSystem), and has been designed to have the following features:

1. Interprocess communication. Programs running under ADEPT can communicate with each other and with programs running elsewhere in the Network. All users can simultaneously have multiple connections. (Presently, there is an overall limit of 32 connections for ADEPT, but this can be increased, if necessary, by enlarging some system tables.)

2. Ability to log in on remote systems. This can be accomplished by writing a user-level program incorporating the TELNET specifications and using interprocess communication. The TELNET specifications will describe the standard, detailed protocol to be used by all nodes in teletypewriter-level (character, text, terminal) communication.

3. Ability to make ADEPT available to remote users. Initially, we will make one of our 10 job entries available to the external ARPA community. HOSTOSS is written in such a way that it can be modified to dynamically vary the number of jobs available. That task will be accomplished next year after system experience is gained and user needs are more clearly identified.

4.1.1 Progress

HOST-to-HOST Protocol Implementation

HOSTOSS has been designed, flowcharted, coded in assembly language, integrated into a new version of ADEPT, and successfully loaded. The non-Network components of the system are running; the Network components are in the process of being debugged. For debugging, we have incorporated into our LISP system a set of Network primitives that allow a LISP user to INITIATE, LISTEN for, ACCEPT, and CLOSE Network connections as well as communicate on open connections. These LISP functions have been thoroughly debugged and are used to invoke the system when we are debugging our Network programs. LISP is a guinea-pig user of the Network at the user level and was chosen because of familiarity, ease of programming, and flexibility while debugging. It also has the desirable side effect of putting our LISP system on the Network early.

The implementation of the HOST-to-HOST protocol has required several system changes in ADEPT to handle necessary buffering, interprocess communications, and control features. These are non-trivial critical changes to the basic ADEPT terminal control architecture. Subsequent protocol changes have, therefore, required extensive recoding and further checkout; although the changes have long-term advantages, their short-term effects have been considerable delay in our implementation phase. The most serious effects were caused by the decision to treat a Network connection as an asynchronous "bit pipe," where the sender must be able to send arbitrary data amounts, with pauses when necessary, and the receiver must be able to collect these fragments into a logical message. This required a redesign of our buffering mechanism. Other system changes were required when the "marking" protocol was abandoned. Network programs in our case are system routines (HOSTOSS has a total of six system components, five in the IBM 360 and one in the DDP-516), and must be thoroughly debugged before they are released for general use. Since SDC management has insisted on high reliability of our time-sharing system,

The Network activity is in the center of the constant battle between maintaining high system reliability and improving system capabilities.

Current debugging indicates that the various HOSTOSS components are properly scheduling each other and are communicating properly through the common tables. In closed-loop tests we have been able to send an INIT command through the IMP (to ourselves) and receive it and the RFNM (acknowledgement) back. We are presently debugging the code that transmits the data during interprocess communication, and expect to have it working soon. This code uses the older protocol with "marking;" flowcharting of the modifications to incorporate the new protocol is in progress.

Several problems encountered with the IMP-HOST interface have also delayed the protocol checkout. BBN is working with us to resolve one aspect of the problem, which involves the apparent loss of messages within the IMP.

The HOST-to-HOST protocol developments and other Network coordination efforts have been pursued by participation in the Network working-group meetings. Also, SDC is represented on the TELNET committee developing the terminal-level protocol specifications. Other activities have included discussions at SRI, BBN, MITRE, and Utah on data base applications of the network.

Distributed Data Base Study

The purpose of the distributed data base study is to investigate techniques for developing a distributed data management system on a computer network, with the ARPA Network specifically in mind. Three approaches were considered:

1. A central data management system, powerful enough to satisfy most users' needs. The system could have distributed data at the various network sites, but the retrieval and update mechanisms would be centralized.

2. Integration of local data management systems and local data (files) through the use of a common data management language and appropriate local interfaces.

3. Development or adoption of a particular data management system to be implemented on all nodes; the retrieval and update language and the logical data structures would be standardized.

After considering these alternatives, we chose approach 2: to investigate the integration of existing data management systems through the use of a common data management language and appropriate local interfaces that would translate the functions described in the common language into the local

data management language. The main advantage of this approach is that it permits the continued use of existing data management systems with existing data bases associated with them, while facilitating the sharing of the data among the network community of users. This approach does not inhibit further development of different local data management systems (which is the case on approach 3). It is anticipated that, under normal use, data will be manipulated locally, for the most part, and less often shared throughout the network. Approach 2 permits continued use of local systems. Other advantages that this approach provides are a non-centralized data management system, which is desirable in case of a failure of one facility (node), and a greater likelihood of acceptance by users because of the use of local systems. As a first step, a few data management systems that are (or will be) available on the ARPA Network were chosen, and will be used as a model for the development of the common language and the local interfaces.

The problems and pitfalls identified so far are:

1. Can a useful common language be defined that will provide a convenient general way of describing functions such as retrievals and updates? Could English (e.g., CONVERSE English) be the common language?

2. How complex will local interfaces turn out to be, and would it be feasible to implement them locally? Can meta-compiling techniques generate a family of translators from CONVERSE-like intermediate language (IL) to local DMS language?

3. Might the extra layers of interfaces cause on-line response to be too slow? Alternatively, can a central node perform English-to-IL-to DMS_i for all nodes?

There is a relation between this work and work being done by others in the ARPA Network. MITRE's study to demonstrate the feasibility of accessing data located at a remote site would help to determine the practicality of sharing data on the Network. CCA's "centralized" data management system with the trillion-bit memory does not conflict with this work, since it can be viewed as another local system. (A possible common area of interest is the development of a common language.) Also, the trillion-bit memory can be used for on-line storage of files to be used by local interfaces (files containing the correlation between data items of different local files). Another related area of work is the development of the "Form Machine." We participated in this committee and are considering a possible use of the Form Machine for transformation of data transmitted between the local interfaces.

4.1.2 Plans

HOST-to-HOST Protocol

Since the HOST-to-HOST protocol is expected to remain in a state of flux for some time, one of our long-term objectives will be to continue the

ADEPT/HOSTOSS evolution to meet the changing requirements. Shorter term objectives include complete debugging of our present implementation--in particular, the interprocess communication and the TELNET functions--and the necessary coordination and user/program interfaces to allow experimental applications of the network during the summer. These experiments are currently being formulated in cooperation with other projects described in this report and with other nodes in the ARPA Network.

Distributed Data Base Study

Our plans for the next half year are to develop a first cut at the use of CONVERSE-like English as the common language and to specify the specific local interfaces involved. These developments will be coordinated with other interested ARPA participants. The end product of these efforts will be a HOST-to-HOST data management protocol to complement those of TELNET, and graphics that are currently under development.

4.2 GRAPH-META

When our graph-meta work began, in 1966, we were concerned with purely syntax-directed compilers. Later, a compiler-writing system was produced that handled local code optimization. Currently, we are attacking the problem of global optimization, i.e., optimization based on graph analysis of the control flow within a procedure. Although work has been done elsewhere in global optimization, we are pioneering its integration into a compiler-writing system to permit automatic generation of code whose quality approaches, if not surpasses, that of handcrafted code. To this end, the Generators language is being extended to facilitate the programming of algorithms involving directed graphs. New operators, data structures, and data types are being added to the language. Statistical studies have been made to determine what type of storage structure is best suited for implementing these language features for the production of practical optimizing compilers.

In addition to its application in compiler optimization, we hope that the new language will be powerful enough to assist in applying graph theory to network theory, communication theory, circuit design, and resource allocation.

Global optimization is highly dependent upon the availability of detailed information concerning communication in computer programs. In particular, questions arise about the flow of control and information, answers to which completely determine the degree to which the program may be improved (optimized). For example, consider the sequence of code:

```
        ...
(1)     a ← b * c
(2)     d ← b
(3)     e ← c * d
        ...
```

It is known that multiplication (*) is a commutative process; it follows that, under ideal conditions, transitivity would identify the formal expressions

```
        b * c ≡ c * b
and     c * d ≡ d * c
```

and, with d = b, allow statement (3) to be replaced by

```
        (3')            e ← a
```

It is necessary, however, that the "ideal" conditions be specified and satisfied before any attempt is made to modify the code. Clearly, an understanding of the edited code is critical. For example, if there exists some path in the program from a definition of the variable c to statement (2) or (3) that does not pass through statement (1), the substitution would be invalid and would produce incorrect results. Nor need this be the best transformation; if the indicated code lies in a portion of the program that is never executed, it would be best to eliminate it altogether.

Matters relating to the safety of a transformation are also of concern. For example, it would appear that any expression, occurring in a loop, that maintained a constant value in the loop should be computed in some portion of the program (i.e. outside the loop) that lay on all possible paths to the loop but with a lower execution frequency. However, consider

```
        do i ← 1 to n by 1
        begin
        if b = 0
           then a(i) ← 0
           else a(i) ← a/b end
        end
```

Here a/b is constant within the loop, and one would be tempted to transform the code to the form

```
        t ← a/b
        do i ← 1 to n by 1
        begin
           if b = 0
              then a(i) ← 0
              else a(i) ← t end
        end
```

But if b = 0, the computation a/b will cause a divide check, a result that would not have occurred in the original code. A safer approach might be

```
if b = 0
   then t ← 0
   else t ← a/b end
do i ← 1 to n by 1
   begin
      a(i) ← t
end
```

The number of special cases requiring analysis is large, and a rigorous mathematical foundation is necessary to produce an optimizing compiler that produces correct optimal code. This is being done with the mathematical theory of graphs applied to program flow. The flow of control in a program can be modeled by a directed graph. An algebraic system may be joined to the graph to provide a basis for analyzing information flow, computational redundancy and invariance, execution frequency and computation efficiency, data dependencies, resource and register allocations, and so forth.

4.2.1 Progress

The major goal of the project for this contract period is to produce an experimental optimizing FORTRAN-IV compiler as a benchmark for evaluating and demonstrating optimization with the graph-manipulator features of the Generators language of META in a practical situation. This compiler will consist of three passes. The first pass has been coded and is in the process of checkout. Design has begun on passes two and three. No coding has been done on the second pass, and very little on the third. These last two passes are discussed in the section on plans.

The theoretical basis for this compiler, and other work at SDC in compiler optimization, is embodied in a formal text rapidly reaching completion this year. Progress is excellent. Part I of the two parts has been completed; Part II is more than half finished.

Benchmark Compiler Pass I

The first pass performs syntax analysis, converts DO loop and Boolean expressions into GOTO's, and allocates storage, taking into account equivalence declarations. The storage-allocation subroutine is especially interesting because it takes only 3/4 of a page of text when written in the Generators language. In the CDC extended FORTRAN, which was written in FORTRAN, this routine takes about six written pages. As a further measure of

the Generators language's power of expression, a number of algorithms involving directed graphs were written and checked out in ALGOL and APL. The algorithm for ordering the nodes of a graph took six written pages in ALGOL, 20 lines in APL, and 12 lines in the Generators language.

New features have been added to the Generators language for use in the optimization pass. These include floating-point arithmetic, which is needed for compile-time evaluation of expressions, and arrays. One-dimensional Boolean arrays have been added to represent definition use vectors. Graphs themselves may be represented as two-dimensional Boolean arrays or as list structures.

Optimization Text: Part I

A treatise entitled "A Mathematical Theory of Global Program Analysis" is in preparation for publication as a textbook this year.* The book is divided into two major sections; the first is a development of the mathematical structures involved in Global Program analysis, and the second is devoted to the applications of this theory to the problem of optimization. The subjects covered are indicated in the table of contents:

Introduction

Part I: Theoretical Foundations

1. Preliminary Notation
2. Weak Ordering Associated with a Graph
3. Dominance, Partitions and Intervals of Graphs
4. Derived Intervals, Reducible and Irreducible Graphs
5. Vertex Ordering Algorithms
6. Lattice Algebra and the Reduction of Irreducible Graphs
7. The Connectivity Matrix and Prime Cycles

Part II: Global Program Optimization

8. Data Flow Analysis, Dependency and Redundancy Equations
9. Constant Subsumption, Common Subexpression Suppression and Code Motion

*Schaefer, Marvin. A Mathematical Theory of Global Program Analysis. SDC document TM-(L)-4602. August 1970.

10. Loop Optimization, Invariant Expression Removal and Reduction in Strength of Operators

11. Safety Considerations

12. Dead Expression Elimination

13. Subroutine Linkages

14. Execution Frequency Considerations

15. Register and Storage Allocation

Bibliography

Appendix I: Graph Theoretic Algorithms in APL/360

The general problem of determining the logical and computational equivalence of two programs is known to be recursively unsolvable, and is not treated in the text. Although the result is of theoretical interest, it is not directly relevant to the problem of optimization because the compiler writer has control over the transformations employed in the optimization. The necessary and sufficient conditions for preserving equivalence are, however, described in the chapter on Safety Considerations.

4.2.2 Plans

The bulk of our activities for the latter half of the contract will be devoted to checking the theoretical basis of our algorithms via the FORTRAN compiler implementation. These activities will focus on the completion of Passes II and III, and draft publication of the optimization text.

Benchmark Compiler Pass II

Global optimization takes place during the second pass of the compiler. It is done on the basis of a directed graph that represents the flow of control of the program. That is why DO loop and Boolean expressions were expanded into control statements during the first pass. Some compiler writers have feared that information would be lost in such a process, but this does not prove to be the case. All of the loop structures that were originally represented by DO's can be recognized from the graph, in addition to loops that the FORTRAN programmer himself coded with GOTO's. Most of the research described in the optimization text will be used to advantage in this pass.

Benchmark Compiler Pass III

The third pass, the code-generation pass, is similar to code generation in locally optimizing compilers. A new algorithm being employed here is the use of a few general registers augmented by core for temporary storage. This means that intermediate results will normally be kept in registers but will be stored when an insufficient number of registers are available. It looks as if it will be possible to program this entire algorithm in the Generators language without adding any special features. This has the advantage of giving the compiler writer control over register allocation rather than freezing a scheme into the compiler-writing system.

Several other problems are being encountered in writing this third (code generation) pass. They are: (1) how to automatically combine library routines into a user's program; (2) what type of subroutine linkage to use; and (3) how to pass addresses to a subroutine, since the current compiler-writing system allows address constants only in common. Several solutions are available for each problem, so it is only an engineering problem of evaluating them and choosing the best.

Optimization Text: Part II

The second part of the optimization text is being written and will be finished during the latter half of the contract. Preliminary drafts will be circulated among experts in the field for scrutiny. A revision based on review comments and feedback from the FORTRAN application will be incorporated into the text. Supplementary appendices containing relevant statistical information may be prepared as time and resources permit.

4.3 STAFF

Systems Research Staff

E. Book, G. D. Cole, Managers

Networks Project

Dr. R. E. Long, Principal Investigator

Dr. A. Shoshani
A. S. Landsberg
Judith C. Needham, NIC Station Agent (part time)

Graph-Meta Project

D. V. Schorre, Principal Investigator

M. Schaefer

4.4 DOCUMENTATION

Long, Robert E. ARPA Network Project--Status and Goals. SDC document N-(L)-24451. February 1971.

Schorre, Dewey V. GRAPH META. SDC document N-(L)-24448. February 1971.

Shoshani, Arie. Distributed Data Base Study. SDC document N-(L)-24452. February 1971.

5. INTERACTIVE SYSTEMS

Interactive systems projects reported here include Problem Solving and Learning by Man-Machine Teams, Time-Sharing, and LISP Extensions, and are part of the larger task of Systems Research covered in section 4. However, they have been broken out for separate treatment here because they all deal with man-machine interfaces, whereas section 4 tasks are focused on internal, machine-machine problems. Furthermore, two of the projects--Time-Sharing and LISP Extensions--are "shadow activities" in that they are mandatory supporting efforts to the more visible research reported earlier.

The Problem Solving project is aimed at building a model, called Gaku, of man-machine cooperation in solving complex, non-deterministic, real-world problems. Successful progress toward that goal is the implementation of a User-Adaptive Language (UAL) for stating to the computer the problems and the interactive steps needed for their solution.

Our time-sharing system, ADEPT, supports the bulk of our research program, and although no research effort is specifically expended for its expansion, development effort is spent to accommodate the research objectives of other projects, most notably Networks, CONVERSE, and Problem Solving. Developments embodied in ADEPT Releases 8.8 and 8.9 include a near-doubling of user-program memory to 85 pages (approximately 348,000 bytes) of core, completion of the Object Sub-System control mechanism, flexible terminal support for a wide variety of baud-rate devices, and a dramatic improvement in system reliability that has resulted in a quadrupling of system mean time to failure.

5.1 PROBLEM SOLVING AND LEARNING BY MAN MACHINE TEAMS

The long-range goal of this project is the development of a man-machine system, called Gaku, that couples the capabilities of man and computer for cooperative planning and creative problem solving in practical, real-world situations. The design of Gaku is a conceptual step toward the realization of a synergistic system--that is, one that combines built-in computer capabilities and human intellectual capabilities to promote the dynamic extension of both kinds of capabilities through man-machine interaction, leading to a "co-evolving" man-machine team.

Previous models of Gaku were designed mainly for preliminary exploration of and experimentation with research ideas and techniques, and were intended for a single user. The past year's accomplishments included (1) a new design of Gaku that incorporates team planning and problem solving and that aims at real-world problems,* (2) the design and detailed specification

---

*Hormann, Aiko M. Planning by Man-Machine Synergism: Characterization of Processes and Environment. SDC document SP-3484. March 1970.

of a User-Adaptive Language (UAL),* and (3) the examination of many real-world problems that are in need of a man-machine approach.

UAL was designed to provide a convenient and flexible means of man-machine communication. It enables the user to begin interacting with Gaku at the initial conceptual stage of problem definition and continue through the exploratory stages of problem solution. UAL is used for two purposes: as a programming language, it is used for initial Gaku implementation; in its extended form, it is used for user-Gaku interaction in problem solving and for designer-Gaku interaction in system modification. Higher-level, user-oriented functions and capabilities can be constructed for the user's convenience through the extensible features of UAL and its techniques for building problem-oriented primitives.

Many areas of potential application were examined, some in detail. Several areas (or classes) of real-world problems to which man-machine techniques may be fruitfully applied have been characterized, and the types of decision dynamics influenced by these characteristics have been identified (Hormann, International Journal of Man-Machine Studies). Military applications in both strategic and tactical planning have also been investigated.

5.1.i Progress

Work during the current contractual period is focused on three related areas--UAL implementation, foundational work toward prototype Gaku development, and problem applications.

UAL Implementation

Prototype UAL is being implemented on the ADEPT time-sharing system, using LISP 1.5 as its source language. Efficiency in the use of both computer time and core memory space was sacrificed in order to speed the implementation. Progress has been rapid since this policy was adopted, and a modestly sophisticated demonstration of man-machine interaction is now possible. UAL's current capabilities include complex list-structure manipulation, function creation, and extensibility (the ability to create new primitives and functions and to define new infix operators). More ambitious UAL features are yet to be implemented.

The major problem encountered early during this reporting period was the limited core memory available in LISP. This problem was alleviated by a new,

---

*Hormann, Aiko; David Crandell, and Antonio Leal. User Adaptive Language (UAL): A Step toward Man-Machine Synergism. SDC document TM-4539. April 1970 (draft).

"growable" LISP system that became available during the period (see section 5.3). However, considerable space in the new system will be consumed by Gaku implementation, leaving little for user-generated programs. Response time is currently tolerable, but will not be when more complex functions for list manipulation are exercised. The impact of these limitations is serious, since relaxation of the constraints is beyond immediate cost and system practicality. A major re-evaluation of the technical approach is in progress.

Foundational Work Toward Prototype Gaku Development

From the many features in the Gaku design, a very basic set has been selected for the first phase of development. These features are being implemented in UAL, in a pencil-and-paper simulation, so that when UAL is ready, basic Gaku can become operational. So far, only rudiments of Gaku's executive, user-portrayal, and graphic-display components have been written in UAL. (The user-portrayal component allows the user to leave unspecified portions that are to be filled in later as on-line decisions.)

Problem Applications

Many problems in the areas of military planning and logistics, law enforcement and criminal justice, health-care programs, and business planning and management have been examined with the following criteria in mind:

1. The problem is sufficiently complex and ill-defined (and unsolved) that the use of man-machine synergistic techniques may open up new possibilities of solving it, or at least of coping with its decision-making intangibles logically and systematically. The man-machine team will be able to examine a much larger number of alternatives, weighing many different factors, than it normally might before a final decision has to be made.

2. The need for good decisions (or answers, or new insights and understanding) is sufficiently urgent, and the possible effects of a wrong (or inadequate) decision are serious enough, that strong arguments can be made for investing the extra time, effort, and funds that would be necessary to employ man-machine synergistic techniques.

3. The problem poses processing requirements that are within the capabilities of the current SDC facilities. The conditions to be met include tolerable memory requirements, maximum tolerable on-line response time (different time limits are likely to be required for different types of interaction), and sophistication in the use of the display-scope techniques.

Although many of the problem situations that have been examined meet the first two criteria, only a few meet the third; most require a large data base, or a large set of problem-oriented programs, or both. Current support levels limit us to attacking either a relatively small, simpler class of problems, or a specific, limited aspect of the problem-solving process. We are pursuing the latter course, as described in the following paragraphs.

One aspect of problem solving that is common to a wide variety of problem-solving situations, and that appears to satisfy the above criteria, is that of the value judgments that enter into evaluating and comparing proposed courses of action (or designs), especially those that affect the public or require extensive funding and other resources. Man's ability to evaluate alternative courses of action grows increasingly tenuous as the number of criteria that must be considered increases. The difficulty is exacerbated when, in group decision making, different value orientations are present and must somehow be reconciled. These value orientations are implicit in both cost and benefit considerations, but there is no technique for defining them with sufficient precision that they can be properly weighed and included in cost/benefit analyses. As a result, important criteria are often excluded from consideration in decision making.

In value judgments, then, we have an element of the problem-solving process that is complex and ill-defined and whose exclusion from the process can result in solutions that are judged either inadequate or wrong. These conditions satisfy the first two of the three criteria listed above. To attack this problem, we have developed a set of methods and techniques (called EVALUATION) that use the "fuzzy-set" concept in the man-machine context (Hormann, SP-3590). Qualitative (or value-oriented) information can be intermixed with quantitative (or factual) information; our techniques will assist evaluators in manipulating qualitative-quantitative information systematically and consistently. The techniques will also assist in group interaction toward a final decision that takes different value orientations into account. Implementing these techniques is not expected to require large amounts of core memory, and they should be able to handle a fair-sized set of data (e.g., 10 alternatives, each with 100 attributes); we expect, then, that the third criterion will be satisfied.

The many potential applications of these techniques that have been identified include evaluations of: (a) complex equipment with many performance criteria (from the buyers' points of view); (b) new products to be introduced into specified segments of the market (from the marketers' points of view); (c) proposals submitted by potential contractors (from the agencies' points of view); (d) proposed locations for large complexes (e.g., missile bases, defense centers, airports, health-care centers, and many more; and (e) proposed configurations of components that interact (e.g., computer hardware with many options for peripheral equipment and software packages).

One concrete application* within area (a) is the selection of appropriate aircraft designs (for a given set of mission requirements) from alternative sets of performance characteristics, including both factual information and value-judgment information.

5.1.2 Plan

Plans for the next six months include continued work in the three areas described--UAL implementation, foundational work toward prototype Gaku development, and problem applications. To alleviate the core-memory problem, UAL programs will be tightened, and techniques for "swapping"segments of UAL and Gaku in and out of disc and tape storage will be explored. Implementation of the EVALUATION tool will be started, working toward creating a demonstrable prototype; other application problems, in addition to the aircraft design example, will also be examined.

5.2 TIME-SHARING

The ADEPT time-sharing executive** functions as the operating system for SDC's ARPA research projects. The system was designed for the IBM 360/50H computer and modified to run on the IBM 360/67I. In the past, a number of system releases were produced for various government agencies, and, during the previous year, the system served as an experimental basis for systems research in time-sharing, networks, natural input/output, and flexible digital communications. Communications research and development activities center on a combined hardware/software system developed to flexibly interface a variety of terminals and display devices to time-sharing systems, ADEPT in particular. The hardware consists of a Honeywell DDP-516 connected through a Honeywell-supplied interface with the multiplexor channel of the IBM 360. The software consists of a Honeywell-supplied programmed multiline controller (PMLC), which was modified and expanded to serve as our operating executive program in the DDP-516.

During the reporting period, in support of our systems research goals, the project was concerned with the hardware/software systems design and implementation activities that relate to the ADEPT-67 and DDP-516 executives.

---

* Suggested by a member of the ARPA/AGILE group.

** Weissman, Clark, Clay E. Fox, and Richard R. Linde. The ADEPT-50 Time-Sharing System. SDC document SP-3344. August 1969.

5.2.1 Progress

ADEPT Transfer

During the reporting period, ADEPT was transferred from the IBM 360/50 to the IBM 360/67 computer (Linde, N-(L)-24460). The transfer involved the utilization of a new swap device--a 2301 drum rather than the Model 50H 2303 drum. However, during this reporting period, through an effort to improve the efficiency of the system, a major problem with the swap device was uncovered. Through the use of our system benchmark programs,* it was found that swaps from the 2301 were 25% slower than those from the 2303 because the timing of the 2301 channel program was slower than anticipated; two dummy records were inserted on each addressable track to resolve the timing limitations. Subsequent analysis has shown that swap speeds are four times that of the Model 50H 2303 drum, which is the expected efficiency gain.

Large Program Support

Because of the increasing demand by users for more memory (see sections 3.1, 4.1, and 5.1), a second IBM 360/67 release began daily operation on 30 November 1970. This release, ADEPT 8.8, was designed to make use of the additional 64 4096-byte pages of core memory available on the 360/67I (Linde, N-(L)-24460). Core was apportioned to give our user programs a greater amount of resident core and to bolt portions of our high-usage swappable executive programs. On the Model 50H, user programs could grow only as large as 46 pages; ADEPT 8.8 will support 85-page user programs.

Network Interface

On 3 February 1971, a third 360/67 release (ADEPT 8.9) was constructed for system testing and experimentation. This release includes the system modules to interface with the ARPA Network (see section 4.1). The Systems Research activity was involved with interfacing these components to ADEPT 8.8. In the main, this interface involved redesigning our batch monitor job (a ghost job in that it has no interactive terminal associated with it) to support an executive Network Control Program and an object process that runs in supervisor state. Other changes involved adding a supervisor-state SVC call and an executive, internal Load-and-Go call to the system.

The following additional changes were made to the 367/67 executive portion of the system:

---

*Karush, A. Benchmark Analysis of Time-Sharing Systems. SDC document SP-3347. June 1969.

- A utility monitor function to process /PRESTORE, /PUNCH, and /PRINT commands in an overlapped mode.
- An I/O error-recording facility for recoverable errors. This printout occurs on the operator's 1052 terminal.
- A utility program for saving and restoring ADEPT file structures on backup tapes. This program is under control of the operator and runs in the supervisor state.
- A statistics program was written to give a measure of the daily operating systems performance.

Communications Multiplexor

Our DDP-516 Programmable Controller--programmed to achieve flexible interfaces--serves as our communications multiplexor. As part of our continual seeking of more cost-effective improvements, we are planning to use an ADS (American Data Systems) multiplexor connected to the synchronous single-line controller on a direct multiplex channel (DMC) to do the bit search and character building now done in the PMLC (see Figure 5-1). This will reduce processor overhead, allowing more--and a greater variety of--simultaneous terminals.

Using a 300-baud clock and a 110-baud clock interrupting at seven times the bit rate, we are now getting 2,870 interrupts per second. This accounts for much of the high (50%) overhead time on the PMLC. If we support 600-baud terminals (which is under serious consideration because of their improved speed and performance), the number of interrupts per second will increase to about 5,000, which is an unacceptable overhead. Using the ADS unit running at 11,040-baud on the DMC, we expect only 60-70 interrupts per second in the DDP-516, and it will not do the character building and bit shuffling now required. The time and storage thus freed permit expansion in both communication channels and baud rates that can be supported.

During the reporting period, a program was written to gain familiarity with the ADS hardware and to perform a diagnostic analysis of the hardware. The routines to interface with the PMLC portion of the DDP-516 executive and the ADS multiplexor were designed, coded, and assembled. These modules have cycled and are currently being debugged.

Hardware problems with the ADS synchronization circuit caused considerable loss of time on the input side of the integration effort. The two ASCII sync characters were inverted, which produced one sync-bit pattern. This problem has been corrected. An error in the Honeywell manual regarding an End-of-Range skip instruction also caused difficulty.

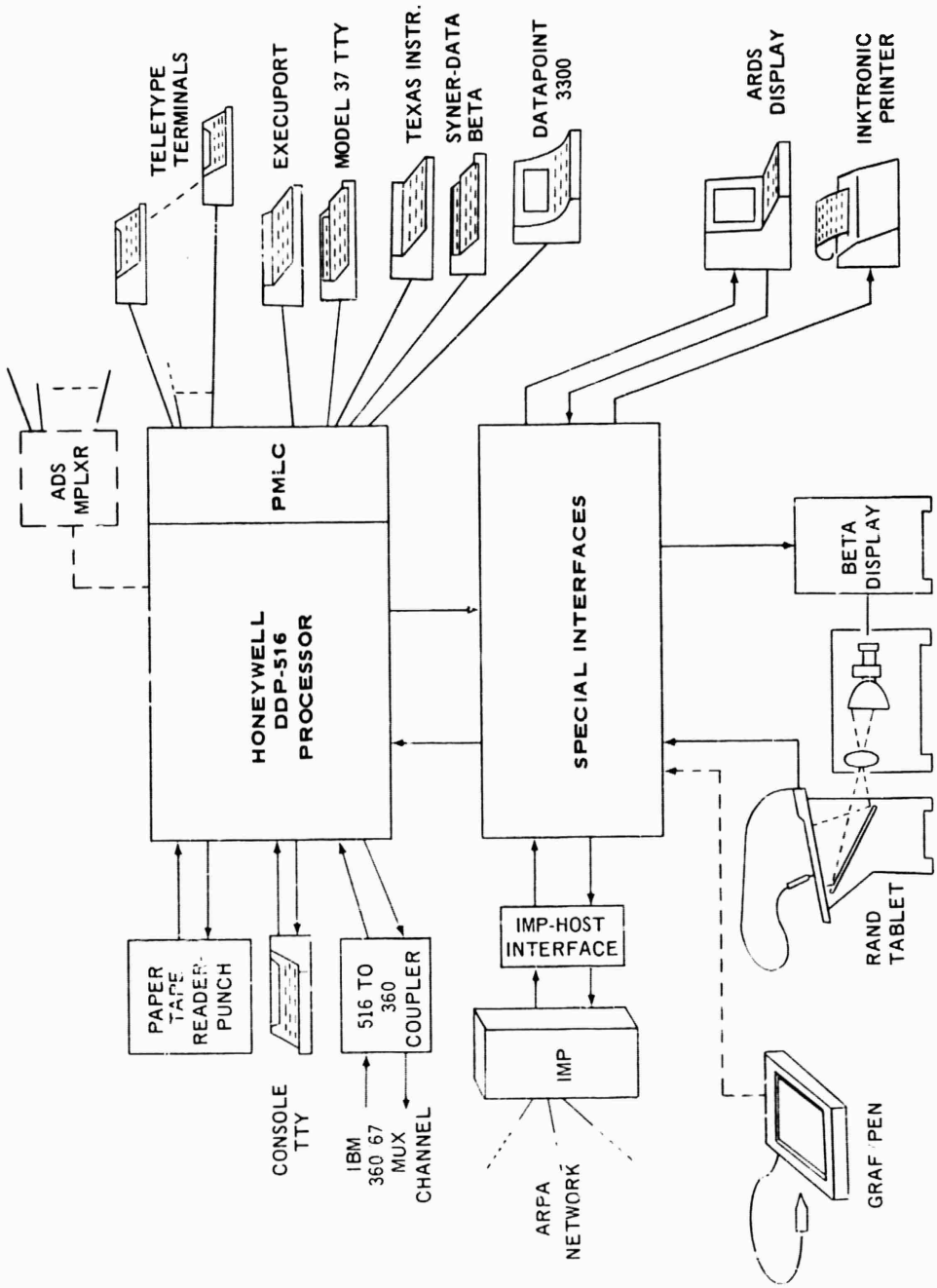


Figure 5-1. Programmable Controller and Associated Equipment

System Reliability

An intensive effort was made to improve the software and hardware reliability of the ADEPT-67 system during this period. One of the more difficult problems occurred between the 360-to-516 hardware and software interface. Communication between the IBM 360 and the DDP-516 can be initiated by either side. However, because of a timing gap between the first signal of initiation from one side and the response from the other side, both sides could initiate communication simultaneously. Although this sequence was anticipated by both the hardware and the software designers, one occurrence of this simultaneous initiation was overlooked by Honeywell. The problem has been resolved, after an extensive investigation of hardware and software logic (Peng, N-(L)-24465), through a one-wire DDP-516 connection.

5.2.2 Plans

During the remainder of the year, a number of activities will take place to extend the capabilities of the ADEPT-67 time-sharing system (Linde, N-(L)-24453). They are as follows:

- An interface to library functions will be implemented with the ADEPT F-level assembler. This will add to the set of MACRO functions available to ADEPT and OS/360 users.

- Completion of the ADS multiplexor system checkout will involve replacement and integration of portions of the Honeywell PMLC with our ADS software interface. The completion of this activity will provide additional core storage for other DDP-516 activities.

- In order to enhance system reliability, it is desirable to add a DDP-516 memory-protection capability. Software utilization of this hardware will follow the ADS system completion.

- An activity to increase the number of ADEPT-67 interactive jobs from 10 to 15 will be initiated this summer upon successful completion of the ADS multiplexor system checkout.

- Completion of basic ARPA Network interfaces will involve additional changes to the ADEPT-67 system. We plan to add more non-conversational jobs to the system for our remote HOST users and to enhance the set of operator Network commands.

5.3 LISP EXTENSIONS

The SDC LISP 1.5 system is a proprietary SDC product written in 1968.* The system operates under the TS/DMS time-sharing executive on IBM 360 computers. SDC has made the LISP system available (on a no-cost basis) to ARPA for many years. It is the basic product that was modified for operation on the ADEPT system. The principal users of the LISP system are the ARPA-sponsored CONVERSE and Problem Solving and Learning by Man-Machine Teams projects (sections 3.1 and 5.1, respectively).

Because of the severe demands made on the LISP system, a number of specific extensions and improvements to LISP are being made. They are:

1. A feature to allow LISP to grow and take advantage of the greater core space made available to users under ADEPT 8.8 (see section 5.2).

2. A TRY and EXIT mechanism to allow LISP programs to field their own errors.

3. A mechanism to allow portions of LISP's binary program space to be unloaded onto disc, giving more space in core for data.

4. An infix translator to extend LISP syntax.

5. Significant improvement to the capabilities and user conventions of LISP interaction, via the time-sharing executive, with peripheral input/output devices.

6. Improvements to the LISP edit program, LISPED.

7. Rewriting the primitive LISP functions CAR, CDR, ATOM, etc., to increase speed and efficiency.

8. Writing a Core Image Generator, which would run under the ADEPT time-sharing executive to allow regeneration of a brand new LISP, thereby gaining more core space by cleaning up certain patch areas. The existence of the Core Image Generator would also make it possible to create specialized copies or versions of the LISP system for special purposes.

9. Creating an expanded set of documents describing the LISP system from the point of view of the user and the system engineer.

---

*Barnett, Jeffrey A., and Robert E. Long. The SDC LISP 1.5 System for IBM 360 Computers. SDC document SP-3043. January 1968.

5.3.1 Progress

The improvements listed in items 1 through 6 above have been written, tested, and checked out to the satisfaction of the principal users. Other ARPA-sponsored projects that occasionally use LISP, such as the Voice Input/Output project (see section 3.3) and the Networks project (see section 4.1), also report satisfaction with the improvements.

With the advent of Release 8.8 of ADEPT, the LISP system could grow from 46 to 85 pages. Although this is only a 46% increase, the LISP system code is now well contained in 36 pages, which means that the amount of data space available to users has quadrupled, from 11.5 to 47 pages. That is a dramatic jump, and has significantly improved LISP performance by lowering the frequency of garbage collection overhead. No further real-memory increases are now possible with SDC LISP, since we are at the limits of physical resources. In attempting future expansions of memory, we will have to explore virtual storage and paging concepts.

No serious technical problems have been encountered in making the improvements listed above. However, users have reported one or two occasions of running out of program reference space. This seems to be caused by the increase in binary program space (for compiling additional code) given the user by both the GROW feature and the feature that swaps binary code to a disc. Each new function takes an additional word in program reference space. This problem will be self correcting if and when the Core Image Generator is written and the entire system thus regenerated.

The LISP Extensions activity is a cooperative effort of ARPA projects. Since work is done by members of SDC's LISP user community, it can continue only at a rate determined by the amount of time the users can spare from their primary tasks.

5.3.2 Plans

Three remaining tasks (7, 8, and 9 above) still remain to be done: improvement of the primitive functions CAR, CDR, ATOM, etc.; Core Image Generation; and documentation. Because of the nature of the staffing of this activity, completion of these tasks is subject to project needs.

5.4 STAFF

Problem Solving and Learning by Man-Machine Teams Project

Aiko M. Hormann, Principal Investigator
A. Leal

Time-Sharing Project*

R. R. Linde, Leader

B. D. Gold
D. M. Gunn
Patricia Kribs
R. H. Larson
A. Tschekaloff

LISP Extensions Project*

J. F. Burger, Leader

J. A. Barnett
A. Leal
Dr. R. E. Long

5.5 DOCUMENTATION

Hormann, Aiko M. "A Man-Machine Synergistic Approach to Planning and Creative Problem Solving: Part I." International Journal of Man-Machine Studies, Vol. 3, No. 2 (1971).

Hormann, Aiko M. Machine-Aided Value Judgments Using Fuzzy-Set Techniques. SDC document SP-3590. March 1971.

Linde, Richard R. ADEPT Extensions Schedule. SDC document N-(L)-24453. February 1971.

Linde, Richard R. Program Specifications for the I Core Version of ADEPT. SDC document N-(L)-24460. March 1971.

Linde, Richard R. Installation of ADEPT on the Model 67 Computer. SDC document N-(L)-24461. March 1971.

Peng, Te-Fu. The Correction of the Attention Bit Problem in the 516/360 Coupler. SDC document N-(L)-24465. March 1971.

*The nature of this task is such that it does not require full-time activity, but the part-time services of a number of people. Only the principals are noted here.